1988-12

Parameter plane analysis of automatic control systems using an IBM compatible microcomputer

Kranz, Richard John, III

NAVAL POSTGRADUATE SCHOOL
Monterey, California

THESIS

PARAMETER PLANE ANALYSIS WITH AN
IBM COMPATIBLE MICROCOMPUTER

by

Richard John Kranz III

December 1988

Thesis Advisor: George J. Thaler

Approved for public release; distribution is unlimited

REPORT DOCUMENTATION PAGE

a. REPORT SECURITY CLASSIFICATION: Unclassified

1b RESTRICTIVE MARKINGS

a. SECURITY CLASSIFICATION AUTHORITY

3. DISTRIBUTION/AVAILABILITY OF REPORT: Approved for public release; Distribution is unlimited

b DECLASSIFICATION/DOWNGRADING SCHEDULE

. PERFORMING ORGANIZATION REPORT NUMBER(S)

5 MONITORING ORGANIZATION REPORT NUMBER(S)

a. NAME OF PERFORMING ORGANIZATION: Naval Postgraduate School

6b OFFICE SYMBOL (If applicable): 62

7a NAME OF MONITORING ORGANIZATION: Naval Postgraduate School

c. ADDRESS (City, State, and ZIP Code): Monterey, California 93943-5000

7b. ADDRESS (City, State, and ZIP Code): Monterey, California 93943-5000

a. NAME OF FUNDING/SPONSORING ORGANIZATION

8b OFFICE SYMBOL (If applicable)

9 PROCUREMENT INSTRUMENT IDENTIFICATION NUMBER

c. ADDRESS (City, State, and ZIP Code)

10 SOURCE OF FUNDING NUMBERS

PROGRAM ELEMENT NO	PROJECT NO	TASK NO	WORK UNIT ACCESSION NO

1. TITLE (Include Security Classification) PARAMETER PLANE ANALYSIS WITH AN IBM COMPATIBLE MICROCOMPUTER

2. PERSONAL AUTHOR(S): KRANZ, Richard J. III

3a. TYPE OF REPORT	13b TIME COVERED	14. DATE OF REPORT (Year, Month, Day)	15 PAGE COUNT
Master's Thesis	FROM ____ TO ____	1988 December	141

6. SUPPLEMENTARY NOTATION The views expressed in this thesis are those of the author and do not reflect the official policy or position of the Department of Defense or the U. S Government.

7 COSATI CODES

FIELD	GROUP	SUB-GROUP

18 SUBJECT TERMS (Continue on reverse if necessary and identify by block number): Parameter Plane, Automatic Feedback Control, Mitrovic's Technique

9. ABSTRACT (Continue on reverse if necessary and identify by block number)

A group of lesser used analog control system design techniques, generally termed parameter plane methods, is examined through the use of an IBM compatible microcomputer program developed as part of this thesis.

The coefficients of a system's characteristic polynomial are determined by the plant and any added compensators. As these coefficients are varied, so too are the roots of the characteristic equation and therefore the system response in terms of bandwidth, settling time, etc.

In the parameter plane method, a designer selects two parameters of a system's compensator(s). The parameters commonly represent such attributes as a compensator gain, pole or zero but can be any linear system function. One or more system characteristics dictating desired system performance, such as relative damping or undamped natural frequency, are computer model inputs. The associated parameter values to achieve the input characteristics are output in graphical and/or tabular form.

20 DISTRIBUTION/AVAILABILITY OF ABSTRACT: ☒ UNCLASSIFIED/UNLIMITED ☐ SAME AS RPT. ☐ DTIC USERS

21. ABSTRACT SECURITY CLASSIFICATION: UNCLASSIFIED

22a NAME OF RESPONSIBLE INDIVIDUAL: THALER, George J.

22b TELEPHONE (Include Area Code): 408-646-2134

22c OFFICE SYMBOL: 62Tr

DD FORM 1473, 84 MAR — 83 APR edition may be used until exhausted. All other editions are obsolete

☆ U.S. Government Printing Office: 1986—606-243

Approved for public release; distribution unlimited.

Parameter Plane Analysis of Automatic Control Systems Using an IBM Compatible Microcomputer

by

Richard J. Kranz III
Lieutenant Commander, United States Navy
B.S., United States Naval Academy, 1974
M.S., University of Southern California, 1982

Submitted in partial fulfillment of the
requirements for the degree of

MASTER OF SCIENCE IN ELECTRICAL ENGINEERING

from the

NAVAL POSTGRADUATE SCHOOL
December 1988

ABSTRACT

A group of lesser used analog control system design techniques, generally termed parameter plane methods, is examined through the use of an IBM compatible microcomputer program developed as part of this thesis.

The coefficients of a system's characteristic polynomial are determined by the plant and any added compensators. As these coefficients are varied, so too are the roots of the characteristic equation and therefore the system response in terms of bandwidth, settling time, etc.

In the parameter plane method, a designer selects two parameters of a system's compensator(s). The parameters commonly represent such attributes as a compensator gain, pole, or zero but can be any linear system function. One or more system characteristics dictating desired system performance, such as relative damping or undamped natural frequency, are computer model inputs. The associated parameter values to achieve the input characteristics are output in graphical and/or tabular form.

TABLE OF CONTENTS

I. INTRODUCTION

A. BACKGROUND

Over the past decade, a resurgence in interest in analog design techniques has been triggered most notably with the advent of the switched capacitor networks. These networks combine analog and digital technology on one chip. Proven analog design techniques which had been eclipsed by digital design methodology have had new life breathed into them. It is widely recognized that they are still valid tools in the design of modern control systems.

This thesis will examine a group of lesser used analog techniques termed parameter plane methods, parameter space methods or Mitrovic's method. This method enables the user to observe changes in the dynamic behavior (e.g., natural frequency, damping ratio, settling time) of the system when certain parameters, such as poles, zeros or gain, are adjusted.

Through the transformation of the differential equations that describe a linear system into algebraic equations in the s-domain, a characteristic polynomial for that system can be obtained. In a feedback control system, the coefficients of this polynomial are determined by the plant and any added compensators. The roots of the characteristic polynomial theoretically determine the

system response in terms of bandwidth, settling time, steady-state accuracy, overshoot, etc. As the coefficients of the characteristic polynomial are varied, so too are the roots and the associated system response characteristics. Two parameters of the characteristic equation α and β, are varied within user defined ranges to produce a plot with these two parameters representing the abscissa and ordinate. In essence, the characteristic polynomial acts as a mapping function whereby s-plane contours are mapped onto the $\alpha - \beta$ plane. One can plot a family of parameter plane curves for various constant values of ζ, ω_n, $\zeta\omega_n$ and/or σ.* By selecting a desired operating point using these curves, the associated α and β values can be graphically determined. The parameter plane plot provides the user with a visual means of deducing how the dominant roots of the characteristic equation move about in the s-plane as the values of α and β are varied.

---

*In a controls system, ζ is the relative damping coefficient, ω_n is the undamped natural frequency and σ is the real part of a root of the system's characteristic equation. The product $\zeta\omega_n$ provides an indication of the settling time of a system, where the settling time is commonly equated to $4/\zeta\omega_n$. All four of these parameters can be precisely determined given a specific characteristic equation root value. Chapter 2 discusses these parameters and Figure 2-2 depicts a root on the s-plane.

B. PURPOSE

There currently exists at the Naval Postgraduate School a parameter plane subprogram of a major controls systems program residing on the school's IBM 370 mainframe. While offering excellent graphics resolution coupled with the availability of other analysis tools, such as the root locus method, in the same controls analysis package, it lacks the portability and ready accessibility inherent with personal computer compatible programs. Current microprocessor capabilities justify the development of a parameter plane program for analysis on a personal computer.

The parameter plane routine developed for this thesis incorporates algorithms which were originally proposed by D. Mitrovic [Ref. 1] and expanded upon and made more versatile by D.D. Siljak [Ref. 2] and G.J. Thaler [Refs. 3 and 4]. These basic algorithms were used by R.M. Nutting in the development of his thesis [Ref. 5] which included a parameter plane program supported by a mainframe computer. Over two decades had passed when D.M. Potter, taking advantage of improved computer codes and vastly superior processors and computer architecture, updated Nutting's program as part of his thesis [Ref. 6] by coding in Fortran 77, simplifying user/machine interactions and utilizing the DISSPLA graphics package available on the school's mainframe.

In addition to the portability and versatility gained through the use of microcomputer compatible programs, the parameter plane program developed in conjunction with this thesis improves user friendliness over existing similar programs. Improvements include the incorporation of menu-driven prompts which provide a user with multiple available options on a single screen. This presentation permits rapid selection of desired functions and plots, including the selection of many commonly desired curve groups with one key stroke. It also allows a user to bypass options not needed on a particular analysis. In addition, the existing mainframe parameter plane routine requires an exceedingly long time to generate and plot the user selected constant curves on the $\alpha - \beta$ plane. This problem is exacerbated when mainframe computer usage load is high. The parameter plane routine presented in this thesis generates and displays plots in significantly less time and with much greater flexibility than was previously possible.

II. PARAMETER PLANE DEVELOPMENT

A. HISTORY

A linear system can be formulated as a single ordinary linear differential equation with constant coefficients. Through the application of the Laplace transform, this differential equation is conveyed from the time domain to the frequency or s-domain and consequently can be manipulated using algebraic techniques. The solution to the algebraic problem of synthesizing a control system was published in a paper by Dusan Mitrovic in 1959 [Ref 1].

Mitrovic's method operates in terms of both the frequency and time domains. It is an analysis and design technique of linear feedback control systems which applies graphical methods to algebraic equations. Mitrovic's procedure is based upon conformal mapping from the s-plane to the coefficient ($\alpha - \beta$) plane through the characteristic equation. This methodology transforms an s-plane presentation into the real domain. The real domain in this case is defined by a coefficient plane whose coordinate axes are two parameters (α and β) of the characteristic equation. These parameters appear in the coefficients of the characteristic polynomial.

In the masters theses of H.H. Chon [Ref. 7] and C.H. Hyon [Ref. 8], Mitrovic's method was applied to a variety

of linear feedback control systems thus illustrating the many useful applications of this technique. However, a major limitation of Mitrovic's method exists in that only two coefficients of the characteristic polynomial can be considered as variables. In real world systems, the desired adjustable system parameters, such as system gain and a pole of a compensator, are often located in more than two coefficients of the characteristic polynomial. Straightforward application of Mitrovic's technique could not be accomplished in such cases.

Recognizing this limitation, D.D. Siljak expanded upon Mitrovic's work by introducing Chebyshev functions in the graphical procedure of the method in 1964 [Ref. 2:pp. 451-453]. This addition simplified Mitrovic's procedure and made it more convenient for computer simulation.

B. BASIC ALGEBRAIC RELATIONSHIPS

The characteristic equation of a feedback control system is simply the denominator of the closed loop transfer function of that system. Given a simple unity feedback system as presented in Figure (2-1), the closed loop transfer function is defined as:

$$\frac{C(s)}{R(s)} = \frac{N(s)}{N(s) + D(s)} = \frac{p(s)}{f(s)} \qquad (2\text{-}1)$$

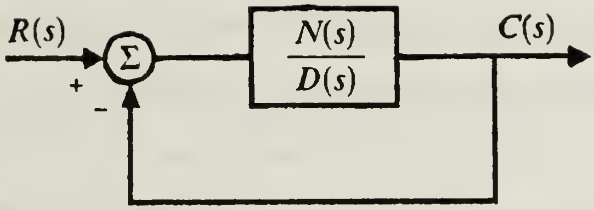

Figure 2-1
Simple Unity Feedback System

$p(s)$ and $f(s)$ are polynomials in the s-domain. The characteristic equation of this system is $f(s)$ and is of the form:

$$f(s) = s^n + a_{n-1}s^{n-1} + a_{n-2}s^{n-2} + \dots + a_2 s^2 + a_1 s + a_0 \qquad (2-2)$$

This form occurs after dividing all terms of the equation by the highest order coefficient (coefficient of s^n).

Mitrovic illustrates the transformation of a characteristic polynomial in the s-domain into one which is a function of ω_n(the undamped natural frequency) and ζ(the relative damping coefficient) by assigning:

$$s = \omega_n \varepsilon^{j(\frac{\pi}{2}+\theta)} = -\omega_n \sin\theta + j\omega_n \cos\theta = -\omega_n\zeta + j\omega_n\sqrt{1-\zeta^2} \qquad (2-3)$$

where

$$0 \le \theta \le \frac{\pi}{2} \quad \text{and} \quad 0 \le \zeta \le 1$$

The original lowest order coefficients of the s-domain characteristic equation are then defined as:

$$a_1 = a_2\phi_2(\zeta)\omega_n + a_3\phi_3(\zeta)\omega_n{}^2 + \ldots + a_n\phi_n(\zeta)\omega_n{}^{n-1} \tag{2-4}$$

$$a_0 = -\omega_n{}^2[a_2\phi_1(\zeta) + a_3\phi_2(\zeta)\omega_n + \ldots + a_n\phi_{n-1}(\zeta)\omega_n{}^{n-2}] \tag{2-5}$$

Functions $\phi_k(\zeta)$ are fixed values regardless of the degree or coefficient values of the characteristic equation. $\phi_k(\zeta)$ is calculated by:

$$\phi_k(\zeta) = -[2\zeta\phi_{k-1}(\zeta) + \phi_{k-2}(\zeta)] \tag{2-6}$$

with

$$\phi_0(\zeta) = 0 \text{ and } \phi_1(\zeta) = -1$$

A plot can be constructed, with parameters a_0 and a_1 as coordinate axes, by calculating the values of a_0 and a_1 as ζ and ω_n are varied over a user defined range. This method obviously lends itself well to digital computer solution.

Analysis of stability and system compensation are important applications with this technique. Through the selection of particular values for a_0 and a_1 corresponding to desired ζ and/or ω_n values, the transient response of the system can be molded.

The ζ equals zero curve corresponds to the imaginary axis of the s-plane. This is the critical line determining system stability. By examining the ζ equals zero curve on

the parameter plane defined by a_0 and a_1, values can be determined that yield a stable system.

As previously stated, Siljak introduced Chebyshev functions in the graphical procedure of Mitrovic's method. As did Mitrovic, he expressed s in terms of ζ and ω_n as depicted in Equation (2-3). The relationship of ζ and ω_n in the s-plane is illustrated in Figure 2-2.

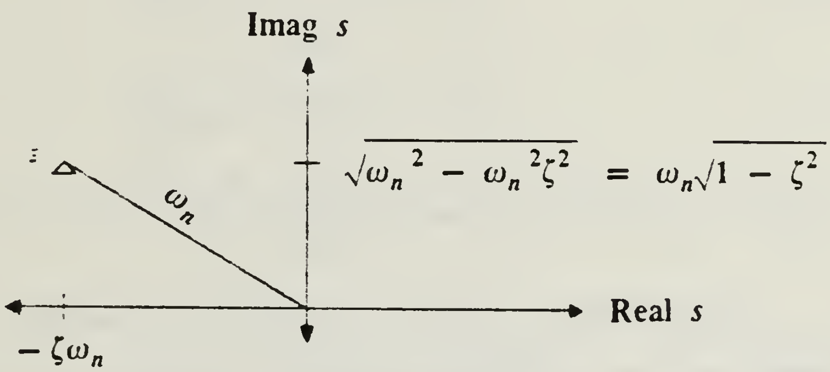

Figure 2-2
S-Plane

$$s = -\omega_n\zeta + j\omega_n\sqrt{1 - \zeta^2}$$

He then applied Chebyshev functions $T_k(\zeta)$ and $U_k(\zeta)$ to transform Equation (2-3) into:

$$s^k = \omega_n{}^k[T_k(-\zeta) + j\sqrt{1-\zeta^2}\,U_k(-\zeta)] \quad (2\text{-}7)$$

where

$$T_k(-\zeta) = (-1)^k T_k(\zeta) \quad (2\text{-}8)$$

$$U_k(-\zeta) = (-1)^{k+1} U_k(\zeta) \quad (2\text{-}9)$$

The argument ζ of the Chebyshev functions is restricted to $0 \le \zeta \le 1$ for stable systems.

The functions $T_k(\zeta)$ and $U_k(\zeta)$ can be calculated through application of the recurrence formulas:

$$T_{k+1}(\zeta) - 2\zeta T_k(\zeta) + T_{k-1}(\zeta) = 0 \tag{2-10}$$

$$U_{k+1}(\zeta) - 2\zeta U_k(\zeta) + U_{k-1}(\zeta) = 0 \tag{2-11}$$

where

$$T_0(\zeta) \equiv 1$$
$$T_1(\zeta) \equiv \zeta$$
$$U_0(\zeta) \equiv 0$$
$$U_1(\zeta) \equiv 1$$

Alternately, the Chebyshev function values can be computed using trigonometric functions as follows:

$$T_k(\zeta) = \cos(k\, arccos\, \zeta) \tag{2-12}$$

$$U_k(\zeta) = \frac{\sin(k\, arccos\, \zeta)}{\sin(arccos\, \zeta)} \tag{2-13}$$

Siljak then substitutes the value of s derived in Equation (2-7) into the equation for a characteristic polynomial as shown in Equation (2-2). By applying the condition that the summation of the real terms and the imaginary terms must go to zero independently, the characteristic polynomial can be rewritten as two simultaneous equations:

$$\sum_{k=0}^{m} a_k \omega_n{}^k T_k(-\zeta) = 0 \qquad (2\text{-}14)$$

$$\sum_{k=0}^{m} a_k \omega_n{}^k U_k(-\zeta) = 0 \qquad (2\text{-}15)$$

The function $T_k(\zeta)$ may be expressed in terms of $U_k(\zeta)$ as follows:

$$T_k(\zeta) = \zeta U_k(\zeta) - U_{k-1}(\zeta) \qquad (2\text{-}16)$$

Redefining $T_k(-\zeta)$ and $U_k(-\zeta)$ in terms of Equations (2-8) and (2-9) yields:

$$\sum_{k=0}^{m} (-1)^k a_k \omega_n{}^k U_{k-1}(-\zeta) = 0 \qquad (2\text{-}17)$$

$$\sum_{k=0}^{m} (-1)^k a_k \omega_n{}^k U_k(-\zeta) = 0 \qquad (2\text{-}18)$$

The coefficients of the characteristic polynomial can be divided into α, β and constant terms in this manner:

$$a_k = b_k \alpha + c_k \beta + d_k \qquad (2\text{-}19)$$

Substituting the preceding equation into Equations (2-17) and (2-18) results in the following simultaneous equations:

$$\alpha B_1(\omega_n, \zeta) + \beta C_1(\omega_n, \zeta) + D_1(\omega_n, \zeta) = 0 \tag{2-20}$$

$$\alpha B_2(\omega_n, \zeta) + \beta C_2(\omega_n, \zeta) + D_2(\omega_n, \zeta) = 0 \tag{2-21}$$

where

$$B_1 = \sum_{k=0}^{m} (-1)^k b_k \omega_n{}^k U_{k-1}$$

$$B_2 = \sum_{k=0}^{m} (-1)^k b_k \omega_n{}^k U_k$$

$$C_1 = \sum_{k=0}^{m} (-1)^k c_k \omega_n{}^k U_{k-1}$$

$$C_2 = \sum_{k=0}^{m} (-1)^k c_k \omega_n{}^k U_k$$

$$D_1 = \sum_{k=0}^{m} (-1)^k d_k \omega_n{}^k U_{k-1}$$

$$D_2 = \sum_{k=0}^{m} (-1)^k d_k \omega_n{}^k U_k \tag{2-22}$$

Cramer's rule can be applied to the solution of the two simultaneous equations defined in Equations (2-20) and (2-21):

$$\alpha = \frac{C_1 D_2 - C_2 D_1}{B_1 C_2 - B_2 C_1} \tag{2-23}$$

$$\beta = \frac{B_2 D_1 - B_1 D_2}{B_1 C_2 - B_2 C_1} \tag{2-24}$$

Holding ζ, ω_n or $\zeta\omega_n$ constant and solving the preceding equations while varying the other parameters yields a loci

of points corresponding to the roots of the characteristic equation with constant relative damping, undamped natural frequency or settling time.

However, if adjustments of α and β to achieve a settling time associated with a particular $\zeta\omega_n$ value are of primary concern, Equation (2-3) should be rewritten as:

$$s^k = P_k(\zeta\omega_n, \omega_n^2) + j\omega_n\sqrt{1-\zeta^2}\,Q_k(\zeta\omega_n, \omega_n^2) \tag{2-25}$$

The functions P_k and Q_k are related to the Chebyshev functions T_k and U_k as follows:

$$P_k(\zeta\omega_n, \omega_n^2) = \omega_n^{\,k}T_k(-\zeta) = (-1)^k\omega_n^{\,k}T_k(\zeta) \tag{2-26}$$

$$Q_k(\zeta\omega_n, \omega_n^2) = \omega_n^{\,k-1}U_k(-\zeta) = (-1)^{k+1}\omega_n^{\,k-1}U_k(\zeta) \tag{2-27}$$

The recurrence formulae associated with P_k and Q_k are:

$$P_{k+1} + 2\zeta\omega_n P_k + \omega_n^{\,2}P_{k-1} = 0 \tag{2-28}$$

$$Q_{k+1} + 2\zeta\omega_n Q_k + \omega_n^{\,2}Q_{k-1} = 0 \tag{2-29}$$

where

$$P_0(\zeta\omega_n, \omega_n^2) \equiv 1$$

$$P_1(\zeta\omega_n, \omega_n^2) \equiv -\zeta\omega_n$$

$$Q_0(\zeta\omega_n, \omega_n^2) \equiv 0$$

$$Q_1(\zeta\omega_n, \omega_n^2) \equiv 1$$

As with T_k and U_k, P_k can be expressed in terms of Q_k:

$$P_k = -\zeta\omega_n Q_k - \omega_n^2 Q_{k-1} \qquad (2\text{-}30)$$

In the same manner as Equations (2-17) and (2-18) were derived, Siljak produced two simultaneous equations in terms of Q_k:

$$\sum_{k=0}^{m} a_k Q_{k-1} = 0 \qquad (2\text{-}31)$$

$$\sum_{k=0}^{m} a_k Q_k = 0 \qquad (2\text{-}32)$$

Once again, as in Equation (2-19), a_k is made up of α, β and constant terms. Therefore, the solutions to α and β in this case are identical to Equations (2-23) and (2-24). However, the expressions for B_1, B_2, C_1, C_2, D_1 and D_2 now become:

$$B_1 = \sum_{k=0}^{m} b_k Q_{k-1}$$

$$B_2 = \sum_{k=0}^{m} b_k Q_k$$

$$C_1 = \sum_{k=0}^{m} c_k Q_{k-1}$$

$$C_2 = \sum_{k=0}^{m} c_k Q_k$$

$$D_1 = \sum_{k=0}^{m} d_k Q_{k-1}$$

$$D_2 = \sum_{k=0}^{m} d_k Q_k \qquad (2\text{-}33)$$

As with functions $\phi_k(\zeta)$, both Chebyshev functions are fixed values, not affected by differences in coefficient values or the degree of the characteristic polynomial. Once again, solution of Mitrovic's method incorporating Chebyshev functions lends itself nicely to digital computer techniques.

III. PARAMETER PLANE PROGRAM DESCRIPTION

A. BACKGROUND

1. Introduction

The parameter plane program is menu driven whenever possible. Many menus have an option to select other menus and these, in turn, may have that same feature. If the program user does not wish to examine particular curves, printer or labeling options, roots, etc., he is not subject to stepping through unwanted menus or responses.

2. Software/Hardware

The programming language used in coding this program is Microsoft FORTRAN77 V4.01. The Plotworks PLOT88 graphics library is used to generate output plots.

All simulations conducted for this thesis were performed on IBM-AT or IBM compatible 80286 machines. Due to the universality of FORTRAN coding, this program could be implemented on any machine capable of being programmed in FORTRAN. The source code for the parameter plane program is listed in Appendix.

A note before running the program. ANSI.SYS should be incorporated in the personal computer's CONFIG.SYS file prior to running the parameter plane program. Through the ANSI.SYS device driver, system calls to clear the screen and position the cursor are enabled. If not, screen

readability suffers although the program is still fully functional. More will be said on this in the ANSI Module section of this chapter.

3. <u>Parameter Plane Program</u>

The parameter plane program itself consists of less than a page of code. Other than assigning default settings to certain variables used throughout the program, its sole function is to serve as a central switchboard to route calls to the eleven major subroutines which make up the parameter plane package.

There is a minor subroutine which is only called when the program is initiated. This subroutine presents an introductory menu entitled 'LOAD/INSERT MENU' (Figure 3-1). It provides options to load a problem from an existing file, input the characteristic polynomial of a system the user wishes to examine, view an example of how to input a characteristic polynomial or quit the program. A flag corresponding to the selected option is set, control is returned to the main program and the selected subroutine is called and executed.

LOAD/INSERT MENU	
OPTION NO.	OPTION
1	LOAD Problem from File
2	INPUT Characteristic Equation
3	EXAMPLE Characteristic Equation Input
9	EXIT to Main Menu

Enter integer number for selection ===>

Figure 3-1
LOAD/INSERT Menu

Upon completion of loading a characteristic polynomial either manually or through a previously created file, a subroutine providing the user with and titled MAIN Menu is called (Figure 3-2). From here, subroutines encompassing all user available options can be selected. These subroutines are grouped into four broad categories.

MAIN MENU	
OPTION NO.	OPTION
1	CURVE Selection Menu
2	PRINTER Selection Menu
3	REVIEW/CHANGE Selections
4	LOAD Problem from File
5	SAVE This Problem
6	INPUT Characteristic Equation
7	PLOT Curves
8	ROOT Finder
9	EXIT Program

Enter integer number for selection ===>

Figure 3-2
MAIN Menu

There are four primary modules associated with the parameter plane routine: a user utilities module, a curve

selection module, a plotting module and a root finding module. In addition, there is a small module consisting of two subroutines; one to clear the screen and the other to position the cursor on the monitor screen. Figure 3-3 is a schematic illustrating the organization and interrelationships of the modules and associated subroutines of the parameter plane package.

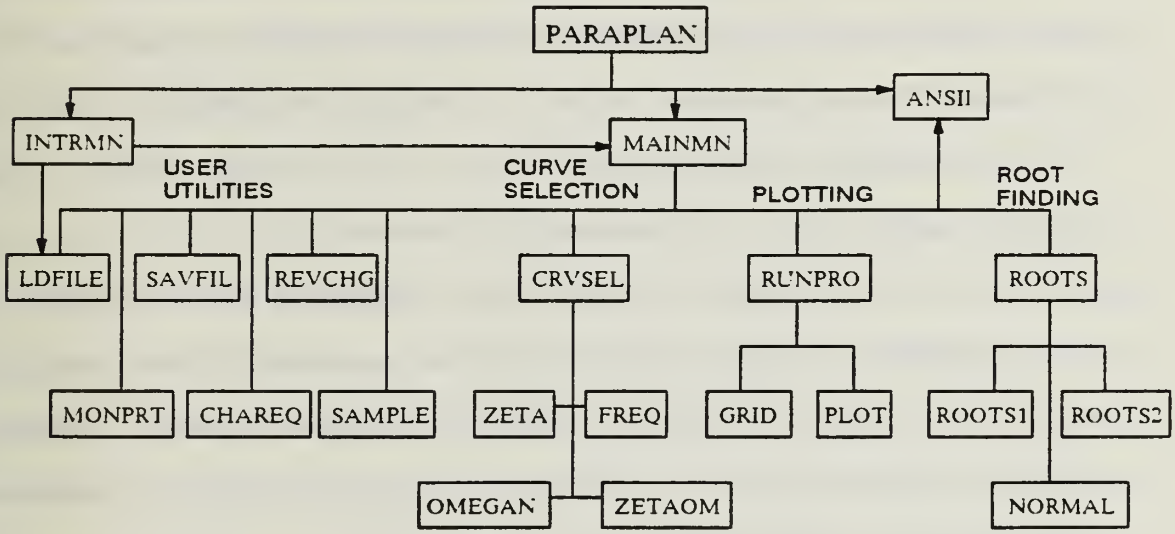

Figure 3-3
Parameter Plane Schematic

B. USER UTILITIES MODULE

1. <u>Overview</u>

The user utilities module contains six subroutines which provide primary call routing procedures, data manipulation and review and output setup. In addition, a sample problem subroutine is incorporated within this module.

2. Subroutine CHAREQ

The provision for manually inputting a characteristic polynomial is provided through Subroutine CHAREQ. The initial prompt displayed upon calling this subroutine requests entry of the order of the characteristic equation. A ninth-order polynomial is the maximum size limitation.

Chapter II goes into great detail explaining the development of the parameter plane method. Equation 2-19 defines the algebraic form of each coefficient of the characteristic polynomial. One can see that each coefficient term is composed of three parts, in this case defined as an alpha, a beta and a constant part. Alpha and beta are the two user defined parameters which the designer wishes to set to obtain certain desired system response characteristics. A coefficient containing only a constant part is not affected when system characteristics such as relative damping or undamped natural frequency are altered. These alterations will of course change the values for alpha and/or beta however. Subsection 6 in this section titled Subroutine SAMPLE provides a simple example on the mechanics of assigning values to the constant, alpha and beta coefficient terms. Chapter IV illustrates solving more complex problems with the parameter plane method.

Following input of system order, requests are made to separately enter the three parts of each characteristic

polynomial coefficient. Constant coefficient terms are entered first. Input is requested from highest order term to lowest. For example, in the case of a third order polynomial, the first prompt would be 'CONSTANT Coefficient of S ** 3 = ' with the cursor positioned just after the equals sign, awaiting user input. In many instances, a polynomial will not contain constant terms for certain coefficients. In these cases, the user must enter a zero for the applicable term(s). When insertion of constant coefficients is complete, they are echoed back and an offer to change incorrectly entered values is provided.

After satisfactory entry of the constant coefficients, the user receives a prompt to enter the alpha coefficient values of the characteristic equation. Once again, these terms apply to the first of two user selected system parameters corresponding to a system or compensator gain, pole, etc. which the user desires to fix in order to achieve a particular set of system characteristics. Chapters I and II discuss the background and development of this method while Chapter IV contains a number of problems which illustrate its application. The same procedures used to enter and verify constant coefficients are used in the alpha value inputs. The user is then prompted to enter the coefficients for the second of two parameters, the beta terms.

When entry of all constant, alpha and beta coefficients is complete, a prompt is made for input of the minimum and maximum undamped natural frequency values over which to calculate all requested relative damping coefficients. Subsection 6 provides an example on determining these values in discussing Subroutine SAMPLE.

A final function of Subroutine CHAREQ, in conjunction with the root finding module, allows one to enter specific values for alpha and beta and then calculate the roots of the existing characteristic equation.

3. Subroutine LDFILE

Subroutine LDFILE permits program input of a preexisting data file. The user is queried as to file name (not to exceed eight characters) and file extension (three characters or less). The code checks for existence of the file within the working subdirectory. If it does not exist, a message is returned stating such and offering the option of entering another file name or returning to the MAIN Menu. Entry of file extension is optional. In fact, a file can be labeled with as little as one character or number.

The data file contains graphics output options (to the monitor or a particular printer type), minimum and maximum frequencies over which to span constant damping coefficient selections, the order of the characteristic polynomial and associated coefficient values. When loaded, all parameter values are displayed on the screen for

verification. It is most easily created using MAIN Menu option 5, 'Save this problem', after an analysis has been run. This option calls Subroutine SAVFIL to be discussed next. A data file could be created using a text editor and aligning the previously stated parameter values in the format identified by the Subroutine LDFILE source code listed in Appendix A. Of course the first method is quicker and less subject to errors.

4. Subroutine SAVFIL

After entering in Subroutine CHAREQ the previously discussed parameters which define a system and the particular range of frequencies of interest, it would be convenient and time efficient to save the system definitive numbers, if one is to look at that same system in the future. Subroutine SAVFIL provides this service. As with Subroutine LDFILE, a prompt is displayed requesting input of file name and file extension. In addition, a check is made to determine if the entered file name plus optional extension already exists. If it does, the option is provided to overwrite the existing file or enter a new file name and extension. The current parameters as outlined in the Subroutine LDFILE subsection are then written to the indicated file. All parameters are also echoed to the screen as a final check for accuracy.

Even high order polynomials require only 300 to 400 bytes of memory, therefore storage of multiple systems

will have little impact on disk space used. Approximately 10,000 high order systems could be stored on a double sided, double density diskette.

5. Subroutine REVCHG

Occasionally, a need arises to review the system coefficient values and frequency range under consideration. This can be done by loading or saving a file or accessing Subroutine FREQ in the Curve Selection Module. In most cases, this procedure can be accomplished more quickly by selecting option 3, 'REVIEW/CHANGE Selection', in the MAIN Menu. This selection calls Subroutine REVCHG. As indicated by the option title, changes to system parameters can be made directly from this subroutine.

6. Subroutine SAMPLE

Subroutine SAMPLE provides a quick tutorial on the derivation and entry of a system's characteristic polynomial. As an example, a universal third order system is presented.

A universal third order system is represented by the characteristic equation $s^3 + As^2 + Bs + 1 = 0$. A and B are the variables defining the two system/compensator parameters to be determined so as to achieve desired system response. This characteristic equation is obtained with a $1/s^3$ plant incorporating both acceleration (s^2) and velocity (s) feedback. A and B (the alpha and beta terms

identified in Equation 2-19) are the respective gains for the two compensators.

To load this characteristic polynomial into the parameter plane model, one would first select option 2 in the introductory LOAD/INSET Menu or option 6 in the MAIN Menu. Each of these option titles is 'INPUT Characteristic Equation'.

A simple examination of the characteristic polynomial shows that the s^2 and s terms have no constant coefficients while the s^3 and constant (1) terms have no alpha (A) or beta (B) terms. Therefore, the response to 'Enter the CONSTANT Coefficient Values of the Characteristic Equation' would be:

```
'CONSTANT Coefficient of s**3 =' 1
'CONSTANT Coefficient of s**2 =' 0
'CONSTANT Coefficient of s**1 =' 0
'CONSTANT Coefficient of s**0 =' 1
```

The response to the alpha and beta coefficient request is:

```
'ALPHA Coefficient of s**3 =' 0
'ALPHA Coefficient of s**2 =' 1
'ALPHA Coefficient of s**1 =' 0
'ALPHA Coefficient of s**0 =' 0

'BETA Coefficient of s**3 =' 0
'BETA Coefficient of s**2 =' 0
'BETA Coefficient of s**1 =' 1
'BETA Coefficient of s**0 =' 0
```

Entry of minimum and maximum values of undamped natural frequency would be dictated by the desired system response characteristics. For example, a settling time (T_s)

under 2 seconds is required. System settling time is approximated by four divided by the product of relative damping coefficient (zeta) and natural undamped frequency ($omega_n$). Since zeta is restricted to values between zero and one, $omega_n$ ranges can be calculated (using minimum value of .1 for zeta). The result is that $omega_n$ can vary between 2 and 20, therefore enter these values as minimum and maximum frequency.

7. Subroutine MONPRT

The largest of the subroutines within the User Utilities Module is Subroutine MONPRT. A number of smaller subroutines are associated with it. Subroutine MONPRT allows selection of the graphics output device, be it the monitor or a wide variety of printers. Graphics output defaults to the monitor when the program is first executed. Graphics output can always be dumped from the screen to a printer, but there may be times when direct output to a printer is desired. All NPS printers are included in the printer output options. In addition, one option permits the user to directly enter the values for IOPORT and MODEL, as outlined in the PLOT88 Manual [Ref. 10].

The first menu displayed when Subroutine MONPRT is called is the PRINTER/OUTPUT Menu (Figure 3-4). Selection of a printer in this menu automatically directs graphics output to the most commonly associated output port, either parallel or serial, for that particular device. However,

usually there are multiple parallel or serial ports to which a printer can be attached. The default printer output setting may not direct plots to the appropriate port. To provide maximum output flexibility, the PRINTER/OUTPUT Menu allows selection of a specific output port through access to another menu.

PRINTER/OUTPUT MENU	
PRINTER NO.	PRINTER
1	Epson FX-80, All
2	Epson FX-100, All
3	Epson MX-100, All
4	Epson RX-80, All
5	Epson MX-80 & IBM Printer
6	HP 7470A Graphics Plotter
7	HP 7475A Graphics Plotter
8	HP 758xB Series Plotters
9	HP 2686A Laser Jet
10	Graphics Monitor (default)
11	HARDWARE Interface Menu
12	Input PLOT88 Values for IOPORT and MODEL
99	EXIT to Main Menu

Enter integer number for selection ===>

Figure 3-4
PRINTER/OUTPUT Menu

The output port selection option is called the HARDWARE Interface Menu (Figure 3-5). This menu is part of a subordinate MONPRT subroutine titled PORT. The options allow selection of one of three parallel ports (LPT1 - LPT3) or either of two serial ports (COM1 - COM2).

HARDWARE INTERFACE MENU	
SELECT NO.	PORT
1	LPT1 Printer Port
2	LPT2 Printer Port
3	LPT3 Printer Port
4	COM1 Serial Port
5	COM2 Serial Port
99	EXIT to Main Menu

Enter integer number for selection ===>

Figure 3-5
HARDWARE INTERFACE Menu

If a serial port is manually selected in the HARDWARE INTERFACE Menu, an associated data transfer (baud) rate must also be assigned. Immediately after selection of serial port options 4 or 5, the BAUD (data transfer) RATE Menu is displayed (Figure 3-6). Transfer of graphics data over a single line is usually time consuming, therefore the highest printer capable transfer rate available (9600 baud) is normally the best selection. A slower transfer rate would only be indicated if improper graphics output is generated.

BAUD (data transfer) RATE MENU	
SELECT NO.	BAUD RATE
1	300
2	1200
3	4800
4	9600

Enter integer number for selection ===>

Figure 3-6
BAUD (data transfer) RATE Menu

Bit by bit transfer of data can also have a check sum (parity) associated with it to provide an internal check of correct data transfer. Following selection of a data transfer rate, the PARITY Menu is displayed. The user has the option of choosing odd, even or no parity.

PARITY MENU	
SELECT NO.	PARITY
1	NO Parity
2	EVEN Parity
3	ODD Parity

Enter integer number for selection ===>

Figure 3-7
PARITY Menu

The PLOT88 Software Library Reference Manual [Ref. 10] contains a number of tables which provide common output settings for many more printers than can be selected in the PRINTER/OUTPUT Menu.

C. CURVE SELECTION MODULE

1. <u>Overview</u>

The Curve Selection Module contains only one subroutine. However, Subroutine CRVSEL is the single largest subroutine in the entire program and contains all of the coding that makes this program a parameter plane analysis tool. Attempts were made to subdivide this large subroutine into many function specific smaller ones. Apparent Fortran compiler limitations due to the overall size of the program frustrated this effort.

Numerous menus in Subroutine CRVSEL permit the user to rapidly select a variety of data display and file options as well as commonly desired constant parameter curves such as the zeta equals zero curve, defining the system stability limit. These menus also allow one to easily determine the curve selection process level and provide the ability to backtrack if necessary.

Three constant parameter choices are offered for solution of alpha and beta values associated with a particular root position and subsequent plotting on the alpha/beta plane. The first of these choices is the constant zeta contour. Zeta is the variable normally assigned to the relative damping coefficient. It is represented on the s-plane as a radial vector extending outward from the origin and is directly related to system steady state overshoot. The zeta equals zero curve is the

imaginary axis of the s-plane while the zeta equals one curve falls on the real axis. The second contour available for plotting is the constant $omega_n$ curve representing system undamped natural frequency. Its s-plane representation is a curve of constant magnitude about the origin. The intersection of these two contours exactly defines a system root location. The final curve is the product of the first two contours and indicates the transient response of the system. This curve plots as a straight line on the s-plane parallel to the imaginary axis. System settling time is usually defined as four divided by the zeta-$omega_n$ product. The interrelationship of these terms with the s-plane is depicted in Figure 2-2.

2. Subroutine CRVSEL

Subroutine CRVSEL is called from the main program through selection of option one in the MAIN Menu, labeled CURVE Selection Menu. Upon selection of this option, the CURVE DATA POINT DISPLAY Menu appears on the screen (Figure 3-8). Option selection from this menu determines if the user is provided with a screen display of every tenth alpha/beta pair and/or all alpha/beta solutions are routed to a data file for future reference. Another available option provides neither a screen display of alpha/beta pairs nor a dump of these data points to file. Selection of this option speeds the curve construction process slightly by avoiding I/O and may be the preferred option if only a

quick graphical overview is desired. The CURVE DATA POINT DISPLAY Menu also provides the option of computing system closed loop root values associated with each alpha/beta pair. These roots can be displayed on the screen and/or written to a file.

CURVE DATA POINT DISPLAY MENU	
OPTION NO.	OUTPUT SELECTION
1	NO output DISPLAY or save to FILE
2	DISPLAY every 10th alpha/beta value
3	DISPLAY alpha/beta values and roots
4	DISPLAY & FILE alpha/betas and roots
5	FILE all alpha/beta values
6	FILE all alpha/beta values and roots

Enter integer number for selection ===>

Figure 3-8
CURVE DATA POINT DISPLAY Menu

When options 4,5 or 6 are made in the CURVE DATA POINT DISPLAY Menu, indicating that calculated data is to be filed, the user is queried as to name and extension of the data file to be opened. A check is made to determine if the entered data file name and extension already exists. If it does, the user is given a choice to overwrite that file with newly generated data or open a new file under another file name.

When the data display and storage procedures just discussed are completed, the primary menu of this subroutine is displayed. It is called the CURVE SELECTION Menu (Figure 3-9) for obvious reasons. In addition to

options allowing selection of any one of the three constant parameter curves previously outlined, options to change the frequency range over which constant zeta contours are calculated, plot the selected curves or return to the MAIN Menu are offered.

```
-------------------------------------------------------------------
|                      CURVE SELECTION MENU                       |
-------------------------------------------------------------------
|  OPTION NO.   |              CURVE SELECTION                    |
-------------------------------------------------------------------
|      1        |     Constant ZETA Curves                        |
|      2        |     Constant OMEGA Curves                       |
|      3        |     Constant ZETA*OMEGA Curves                  |
|      4        |     Change Frequency Range                      |
+++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++
|      5        |     PLOT Selected Curves                        |
+++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++
|      9        |     EXIT to Main Menu                           |
-------------------------------------------------------------------

     Enter integer number for selection ===>
```

Figure 3-9
CURVE SELECTION Menu

Each of the three curve options permits the selection and plotting of up to ten constant contours. In most instances, a designer is interested in viewing the stability limits of a system, as represented by the zeta equals zero curve. Due to this zeta curve attribute and the fact that for a stable system, zeta is restricted to the range zero to one, a CONSTANT ZETA Menu (Figure 3-10) is provided. The user is given the option of entering particular constant zeta curve values or selecting with a single key stroke the zeta equals zero curve or two other options which equally subdivide the range zero to one into three or five parts. Following identification of constant

zeta curves, the program returns to the CURVE SELECTION Menu to allow selection of other constant curves or plot the current selections.

```
-----------------------------------------------------------------
|                     CONSTANT ZETA MENU                        |
-----------------------------------------------------------------
| OPTION NO.  |            ZETA CURVE SELECTION                 |
-----------------------------------------------------------------
|      1      |    Select particular constant ZETA curves      |
|      2      |    ZETA = 0  curve                             |
|      3      |    ZETA = 0,0.5 and 1.0 curves                 |
|      4      |    ZETA = 0,0.25,0.5,0.75,1.0 curves           |
+++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++
|      9      |    EXIT to Curve Selection Menu                |
-----------------------------------------------------------------


     Enter integer number for selection ===>
```

Figure 3-10
CONSTANT ZETA Menu

Each constant curve is calculated using 100 data points. This value was selected as a compromise between speed of curve generation and precision of plot. A possible update to this program would permit a user entered data point number. This would enable a designer to more closely tailor the parameter plane plots to his or her needs.

Successive data points are linearly incremented in the case of constant omega_n and zeta-omega_n curves. In the generation of successive constant zeta points, a logarithmic increment is used due to the large possible variation between minimum and maximum omega_n values. (In calculating alpha/beta pairs associated with zeta curves, omega_n is varied between minimum and maximum while zeta is held constant).

High order systems often exhibit large variations between successive alpha/beta points. Unfortunately, the PLOT88 graphics package occasionally fails when these variations are extreme (i.e., in excess of five orders of magnitude). To prevent inadvertently halting program execution, two checks are made. The first prevents calculation of an alpha or beta value when the denominator of the defining equation is less than 1.e-12. The second compares successive alpha/beta points for a difference exceeding 1.e5. If either situation occurs, a warning is sent to the monitor and, if applicable, to a data file. The offending alpha/beta pair is then assigned the value of the previously calculated point.

A final procedure available in Subroutine CRVSEL is the selection of minimum and maximum frequencies over which constant zeta curves are calculated. Should this option be exercised after selection of constant zeta curves, the program automatically recalculates the associated alpha/beta points.

D. PLOTTING MODULE

1. Overview

The Plotting Module consists of three subroutines. The primary function of this module is to call PLOT88 Graphics Library Routines which generate the constant curve plots on the alpha-beta plane. The alpha/beta arrays are

passed to this module via a common block. This method was adopted when attempts to pass large arrays as arguments of subroutines resulted in stack overflow errors.

As discussed early in this chapter, the latest released Fortran version was used in compiling and linking this program. Microsoft FORTRAN77 V4.01 contains an optimizing compiler that does not directly translate the source code into an object file but instead alters the code to achieve various user defined objectives. Two common optimizations are increased speed and reduced code size. The Plotting Module holds the distinction as the only module that could not take advantage of this optimization. One or more calls to PLOT88 Routines were distorted to the point that program execution was halted unpredictably. Therefore, the optimization functions of the Fortran compiler were bypassed.

2. <u>Subroutine RUNPRO</u>

Subroutine RUNPRO makes up the bulk of this module. When first called, the PLOTTING Menu (Figure 3-11) is displayed. This menu provides the user with a number of plot presentation and labeling options.

OPTION NO.	OPTION
	PLOTTING MENU
1	TITLE output graph and PLOT data
2	PLOT data (no title)
3	SIZE output graph
4	SYMBOL to be plotted at each data point
9	EXIT to Main Menu

Enter integer number for selection ===>

Figure 3-11
PLOTTING Menu

Option 1 permits entry of a title up to 30 characters in length. A routine computes the length of the entered title then centers it at the top of each graph. The third option permits adjustment of plot size to both the screen and the printer. This option was utilized to scale plot outputs for inclusion in this thesis. Another option allows the user to select a particular symbol to be applied to the plot at each data point. Figure 3-12 illustrates some of the symbol choices available.

```
Type in an INTEGER number from 0 through 13 to place a
symbol at each calculated data point. There are 100 data points
for each curve. Examples of symbols with associated numbers:

     2   Triangle
     3   +
     4   X
     8   Z
     9   Y
     11  *
     13  Vertical Line

Enter INTEGER number (0 - 13) ==>
```

Figure 3-12
Symbol Selection Display

A final option permits immediate plotting of data arrays without enhancing the graphical presentation. As with most of the menus, the ability to return to the calling routine, in this case the MAIN Menu, is also available.

Once again, three sets of system parameter curves can be plotted; constant zeta, constant omega$_n$ and constant zeta-omega$_n$. Each set can contain up to ten separate contours. The program sequentially checks for the existence of any curves in each of the three categories. In addition to alpha/beta values, total number of curves and data points within each category are passed to Subroutine RUNPRO from Subroutine CRVSEL. When one or more curves are detected in a particular category, Subroutine PLTCRV is called to actually compute each curve. Requested curves in all categories are first calculated before the first plot is displayed. Sequencing to succeeding plots is accomplished by depressing the Enter key. All zeta contours are drawn as a series of straight line segments one tenth of an inch long. This procedure gives the appearance of a smooth, continuous contour. Omega$_n$ and zeta-omega$_n$ contours are plotted similarly, but with dashed lines made up of different segment lengths.

If both zeta and omega$_n$ curves are present, the last plot consists of both sets of curves. The minimum alpha and beta values and axes increments from the zeta

plot are used to scale this graph. If any omega$_n$ constant contour selection frequencies fall outside the minimum to maximum frequency range designated for constructing the constant zeta curves, these curves may not be displayed within the plotting box. The user can still view the offending contour lines by exiting then reentering the plotting routine and selecting the plot size reduction option in the PLOTTING Menu.

After viewing the requested plots, the user is given the option of focusing on a particular area of the plot. This is accomplished through the EXPAND PLOT Menu (Figure 3-13).

```
 -------------------------------------------------------------
|                      EXPAND PLOT MENU                       |
|-------------------------------------------------------------|
| OPTION NO.  |             EXPANSION SELECTION               |
|-------------------------------------------------------------|
|      1      |     Expand area defined by axes values        |
|      2      |     Expand around a selected point            |
|+++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++|
|      9      |     EXIT plotting routine                     |
|-------------------------------------------------------------|


          Enter integer number for selection ===>
```

Figure 3-13
EXPAND PLOT Menu

A choice is offered to expand a specified area defined by minimum and maximum alpha and beta axes values or expand the plot around a selected point relative to absolute alpha/beta axis positions (e.g., expanding about the center of the plot would be accomplished by entering a value of 3.5 for alpha and 2.5 for beta). The second option

provides the user with a quick look at a particular point however the axes are usually lost with any significant expansion. The first option rescales and displays the alpha and beta axes, therefore it is normally the preferred expansion method.

A few remarks on the mechanics of producing hard copy plots, as they apply to Naval Postgraduate School specific hardware, are in order. Graphics output to a device other than a monitor requires an extended period of time as the data file is loaded into the printer buffer and formatted for hardcopy output. In addition, the user is unable to view the plot when the program has been configured for a non-monitor output device. To avoid these inconveniences, the 'Print Screen' key can be depressed while a desired plot is displayed on the monitor. Printer output is then available in less than a minute. The Laser Jet Printer located in the Controls Lab must be configured as an Epson Printer for the 'Print Screen' function to operate correctly.

3. Subroutine GRID

Subroutine GRID is a short, simple routine called before each plot. It first defines the plot physical origin in relation to the output device. This is usually the monitor screen although it would be a printer page if so defined in Subroutine MONPRT. The origin is presently set at eight tenths of an inch above and to the right of the

output device lower left corner. It cannot be modified by the user.

A box of unmodified dimensions 7" x 5" is drawn from the physical origin, defining the plotting area. In addition, a dashed line grid is inserted at integer intervals of the vertical and horizontal axes. The box, grid lines and contour plots are all size modified (both screen and printer output) through the sizing option of the PLOTTING Menu.

4. Subroutine PLTCRV

This subroutine accepts alpha/beta data arrays and graph title passed by subroutine argument. Common block inputs to size the output plot and place a user identified symbol at each data point are also resident.

An initial call is made to a PLOT88 subroutine specifying axis annotation characteristics such as tic mark placement and axis title height. Next, axes scaling intervals are determined based on alpha and beta value ranges and a respective seven and five unit division of the axes. Additionally, the minimum alpha and beta data point values are identified at this time. The x and y axes minimum and major increment values are displayed on the screen with a message indicating that curve construction is in progress. Meanwhile, a PLOT88 subroutine is called which accepts the 100 data point pairs associated with each constant curve and creates a smoothed contour using the

scaling data previously calculated. Data point symbols are also inserted on each curve if selected by the user in Subroutine RUNPRO. When all contours have been created for a particular parameter (e.g., relative damping coefficient - zeta), control is passed back to Subroutine RUNPRO.

E. ROOT FINDING MODULE

1. Overview

The Root Finding Module is unique among the three major modules of the parameter plane program in that it has no menus associated with it. It is called predominantly from the CRVSEL subroutine of the Main Module when a 'y' response is entered to the question 'Do you want to view the roots of the polynomial?' This question is presented to the user each time calculations for a set of constant curves is requested.

The other option of entering the Root Finding Module is offered in the MAIN Menu as Root Finder. Should the user make this selection, a flag is set, Subroutine ROOTS is called and subsequently the subroutine for inputting a characteristic equation is activated. A choice is given to enter a characteristic equation or find the roots of the current characteristic equation. If the user wishes to enter a new or initial characteristic polynomial, the standard characteristic equation entry procedure is carried out. If not, this portion of Subroutine CHAREQ is

skipped over. In either case, entry of a characteristic equation via the root finding module requires that a user enter specific values for alpha and beta. All associated coefficients of these two variables are multiplied by the respective alpha or beta entries. Subroutine CHAREQ then returns the program to the root finding module for calculation of the input polynomial with specified alpha and beta values. In this manner, one can select particular alpha and beta values and compute all roots of the polynomial with those values.

2. Subroutine ROOTS

ROOTS is the highest level subroutine of the root finding module. Terms defining the order and coefficient values of a polynomial are passed to ROOTS as parameters of the subroutine. Additional parameters passed include the particular alpha and beta values to be used in calculating the coefficients of the polynomial. Another parameter is the flag differentiating between calculation of roots associated with constant curves or the calculation of root values for specific alpha/beta user inputs. A call is made to Subroutine CHAREQ if this flag indicates that the roots of a polynomial associated with a user inputted set of alpha and beta values are to be calculated.

Once the characteristic polynomial is fixed, Subroutine NORMAL is called. This subroutine normalizes the polynomial to ensure that the coefficient of the highest

order term is unity. Finally, Subroutine ROOTS2, which contains the root calculation procedures, is called.

3. Subroutine NORMAL

NORMAL is a subroutine that normalizes the characteristic polynomial. It is also the subroutine which multiplies the alpha and beta coefficient terms by the alpha and beta values passed via subroutine parameters from ROOTS. These two products represent a portion of the coefficient of a particular order term associated with the alpha and beta parameters. They are added to the constant coefficient of that same order term to form the total coefficient of that term. A do loop provides the means for combining the constant, alpha and beta coefficient values into a single coefficient for each term of the polynomial.

When the roots of a polynomial are calculated based upon user input of specific alpha and beta values, Subroutine NORMAL provides another function. In addition to showing the user the roots of the polynomial, the routine also outputs the coefficient values of the polynomial based upon the specific alpha/beta inputs.

4. Subroutine ROOTS1

ROOTS1 is a short subroutine that models the quadratic equation. It calculates the roots of a second order polynomial. This subroutine is called only from the ROOTS2 subroutine after the initial polynomial has been reduced to one of second order.

5. Subroutine ROOTS2

Subroutine ROOTS2 is the heart of the root finding module. The algorithm used in this subroutine iteratively searches for a quadratic factor of the input polynomial. When this factor is found, the original polynomial is reduced by an order of two. This process is repeated until the remainder is a first or second order polynomial. Each time a quadratic factor is found, Subroutine ROOTS1 is called to calculate the roots.

There are two terms in the algorithm that affect both the speed of computation and the accuracy of the solution. The first, labeled EPSLON, sets an acceptable accuracy level for the computed roots. The numerical procedure applies ever decreasing correction factors to the constant and first order terms of the quadratic factor, it is calculating. When the sum of these correction factors is less than an EPSLON value of .00005, the refinement is halted and the roots of the quadratic factor are solved through a call to the ROOTS1 subroutine. In the vast majority of cases, the quadratic factor is determined after relatively few iterations. However, there are cases, especially when the characteristic polynomial contains multiple identical real roots, when the sum of the correction factors is not reduced below EPSLON even after 1000 iterations. If this situation occurs during computation and the routine determines that more than 500

iterations have taken place in determining a single quadratic factor, the procedure is halted and control returned to the calling routine. Should this happen during the computation of roots for multiple alpha/beta values, the alpha and beta values will still be displayed without the associated roots and an attempt to determine roots for the succeeding discrete alpha/beta pair will commence.

The methodology of continually reducing a polynomial by a quadratic factor until all roots are determined is known as Bairstow's method. R.L. Wood incorporated this method in his thesis when he coded a routine which included control systems design through root locus techniques [Ref. 9:pp. 42-43].

F. ANSI MODULE

The ANSI Module contains two subroutines. Subroutine CLR clears the monitor screen. It is used primarily as a precursor to displaying a menu or positioning queries or informational text. Subroutine PSIT positions the cursor immediately after a request for response. This subroutine serves not only to give a pleasing screen display but also provide the user with an immediate reference as to where he is in the program and if an input is required of the user. Both routines can be found in the PLOT88 Reference Manual [Ref. 10:pp. 85-86].

The two subroutines which comprise the ANSI Module make use of ANSI escape sequences. These sequences are a series of characters that can be used to define functions to MS-DOS. In addition to clearing the screen and positioning, moving and saving the cursor's position, ANSI escape sequences can control screen graphics and reassign key definitions.

In order to utilize the escape sequences, the microprocessor must first be able to interpret the commands. It does this through a systems file entitled ANSI.SYS. ANSI.SYS must be defined as a device in the CONFIG.SYS file to enable proper interpretation of ANSI escape sequences (e.g., device=\bin\ansi.sys where \bin\ is the path to ansi.sys). If ANSI.SYS is not resident in the CONFIG.SYS file, the parameter plane program will not clear the screen or position the cursor when so directed. Instead, the indicated escape code sequence will be printed on the monitor whenever a call is made to CLR or PSIT. Although program readability is degraded, parameter plane will still function correctly.

IV. COMPUTER AIDED PARAMETER PLANE ANALYSIS

A. OVERVIEW

An algebraic solution of feedback compensated control systems is usually performed by selecting particular zeta and omegan values that provide desired system response characteristics. A pair of complex conjugate roots are thus placed at specific s-plane locations that will provide the designed for response, if these roots are dominant. A check must therefore be made to satisfy the dominance question. If the positioned roots are not dominant, the design scheme must be altered.

Design procedures which select values of zeta and omegan to position a pair of complex roots in a position of dominance assume that the contributions of all additional roots can be neglected. Although this is certainly not always the case, applying this simplification permits modeling of complex systems with second order polynomials. As such, the following system response parameter equations can be utilized to derive zeta and omegan values which produce desired system behavior:

Maximum Overshoot (step input)

$$M_{pt} = 1 + e^{\frac{-\pi\zeta}{\sqrt{1-\zeta^2}}} \tag{4-1}$$

Settling Time (4 time constants)

$$T_s = \frac{4}{\zeta\omega_n} \tag{4-2}$$

Time of Maximum Overshoot

$$t_p = \frac{\pi}{\omega_n\sqrt{1-\zeta^2}} \tag{4-3}$$

Bandwidth (response magnitude > .707*input)

$$\omega_b = \omega_n\sqrt{1-2\zeta^2+\sqrt{2-4\zeta^2+4\zeta^4}} \tag{4-4}$$

Resonance Peak

$$\omega_r = \omega_n\sqrt{1-2\zeta^2} \tag{4-5}$$

Transient Oscillating Frequency

$$\omega_t = \omega_n\sqrt{1-\zeta^2} \tag{4-6}$$

Phase Margin

$$\Phi_m = \tan^{-1}\frac{2\zeta}{\sqrt{-2\zeta^2+\sqrt{4\zeta^4+1}}} \tag{4-7}$$

A graphical solution using the parameter plane technique, with reference to the resident root finding module, provides a designer with a rapid means of determining parameter values satisfying design requirements while ensuring both stability and root dominance. When a family of curves is computed and displayed, the flexibility

of choosing a number of parameter value pairs which may force design compliance is offered.

B. PROBLEM SOLUTIONS

1. Example 4-1

Figure 4-1 illustrates an ideal instrument servo which is to be damped by velocity feedback and a cascaded gain amplifier. The common single parameter root locus method could easily be employed in the solution of this problem by deleting the gain amplifier. However, this stage provides the designer with increased flexibility and illustrates the ease of parameter plane solution in low order systems, despite an additional parameter.

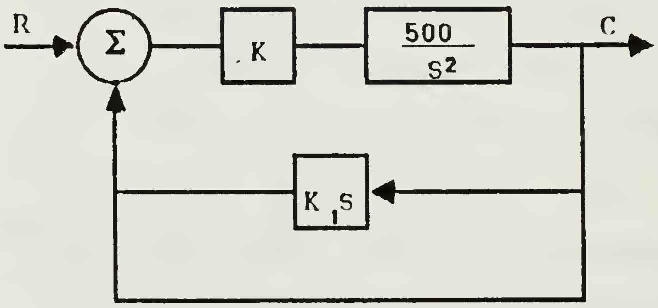

Figure 4-1
Ideal Instrument Servo with Velocity Feedback

Design requirements specify:

Maximum Overshoot Mpt (step input) < 20%

Settling Time Ts < 2 sec

Applying Equations (4-1) and (4-2) with design specifications to solve for zeta and omegan yields:

$$\zeta \geq .46 \tag{4-8}$$

$$\omega_n \geq 4.35\ rad/s \tag{4-9}$$

Let:

$$\alpha = KK_1 \tag{4-10}$$

$$\beta = K \tag{4-11}$$

The characteristic equation for this system is then:

$$s^2 + 500\alpha s + 500\beta = 0 \tag{4-12}$$

Run the parameter plane program by typing in the executable file name, PARAPLAN, when in the subdirectory under which it resides. The IBM microcomputers in the NPS Controls Lab are set up under the file management system 1DIR. To execute the program under this system, enter the appropriate subdirectory by using the up/down arrow keys to position the cursor adjacent to the subdirectory name then depress the ENTER key. Use this same procedure to select PARAPLAN.EXE and push ENTER to run the program.

The LOAD/INSERT Menu will appear. Select Option 2, INPUT Characteristic Equation. Enter coefficient array values after each prompt as follows:

```
'CONSTANT Coefficient of s**2 =' 1
'CONSTANT Coefficient of s**1 =' 0
'CONSTANT Coefficient of s**0 =' 0

'ALPHA Coefficient of s**2 ='    0
'ALPHA Coefficient of s**1 ='  500
'ALPHA Coefficient of s**0 ='    0

'BETA Coefficient of s**2 ='     0
'BETA Coefficient of s**1 ='     0
'BETA Coefficient of s**0 ='   500
```

As this is the first example, let's view the parameter plane over the entire range of zeta. After entering the coefficient array, the program sequences to MAIN Menu. Select Option 1, CURVE Selection Menu. Choose any of the curve data point display options as the next menu appears. Option 1 of the CURVE Selection Menu allows selection of constant zeta curves while Options 2 and 3 provide this service for constant omegan and constant zeta-omegan curves. Input a range of zeta values from zero through one and a range of omegan values from one through ten. Input a value of 2 for the zeta-omegan curve. Select Option 5 to plot the designated curves. The PLOTTING Menu now appears. Either Option 1 or 2 will start the constant curve plotting calculations and subsequently display the plots. The following graphs were obtained by selecting Option 1 and titling the curves.

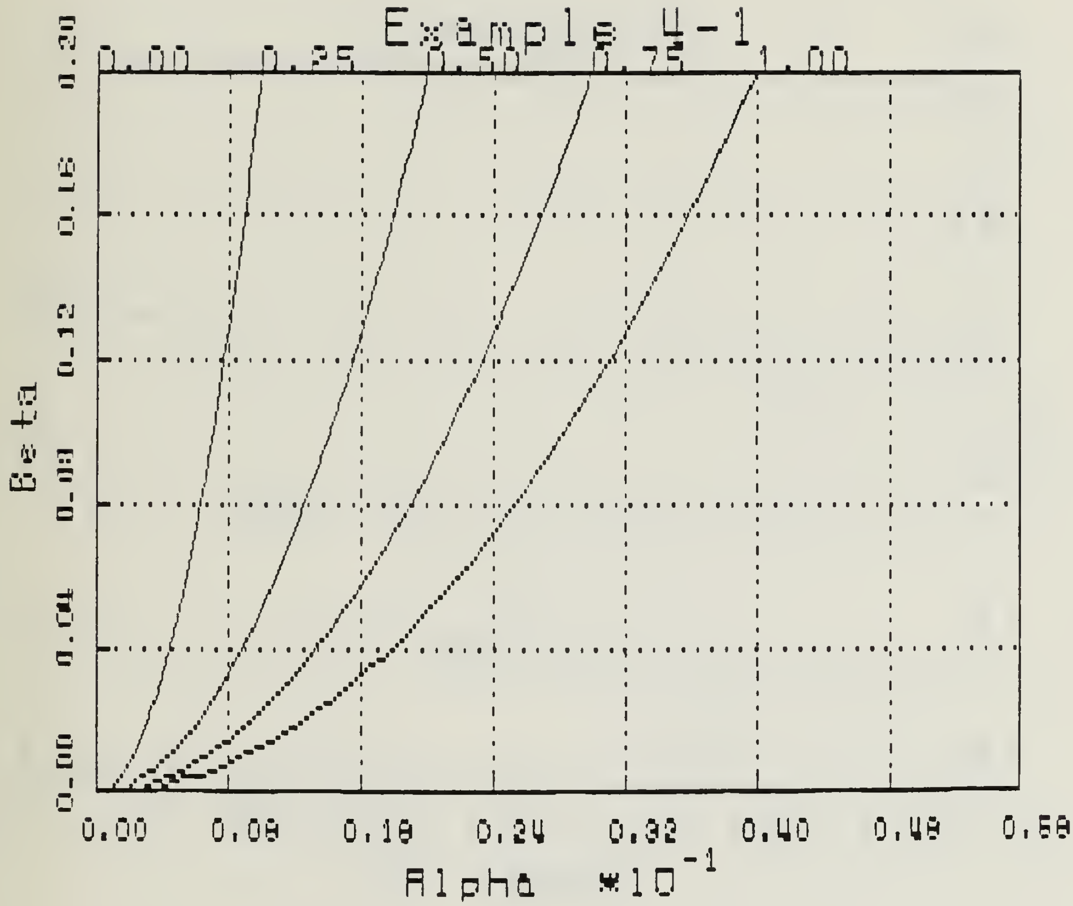

Figure 4-2
Constant Zeta Curves

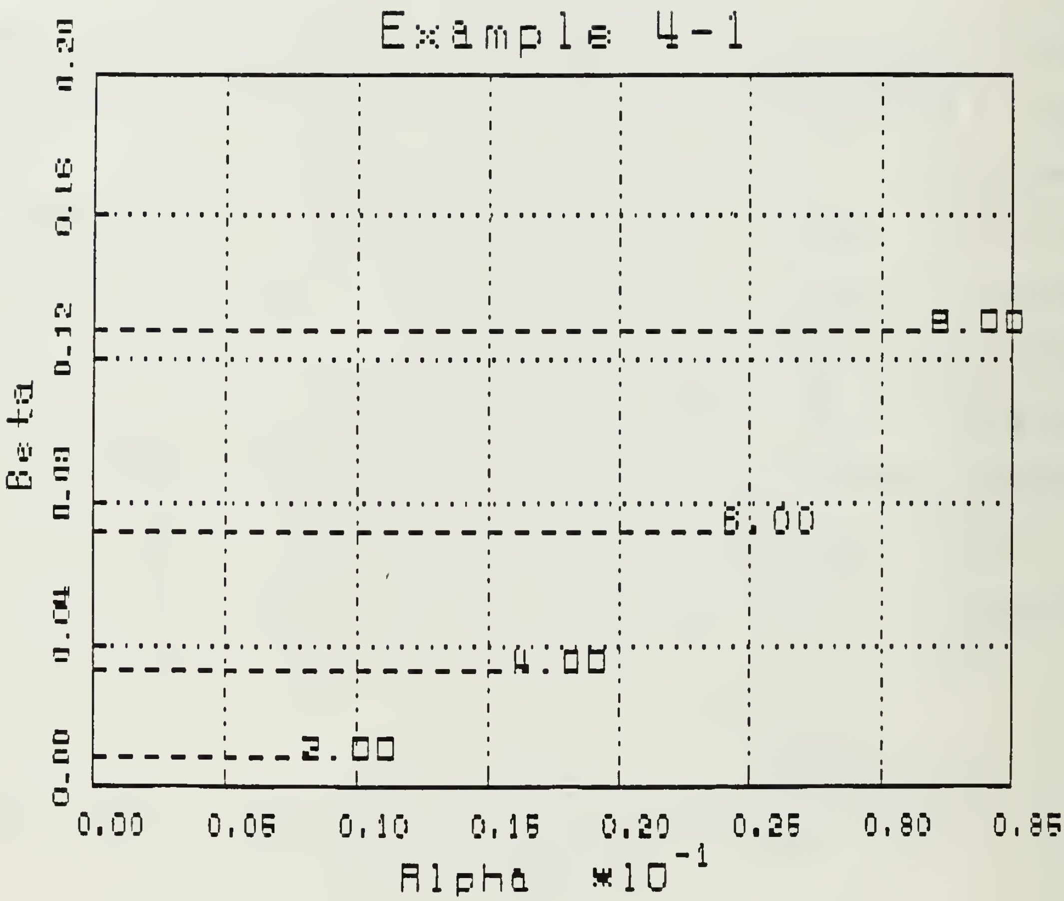

Figure 4-3
Constant Omegan Curves

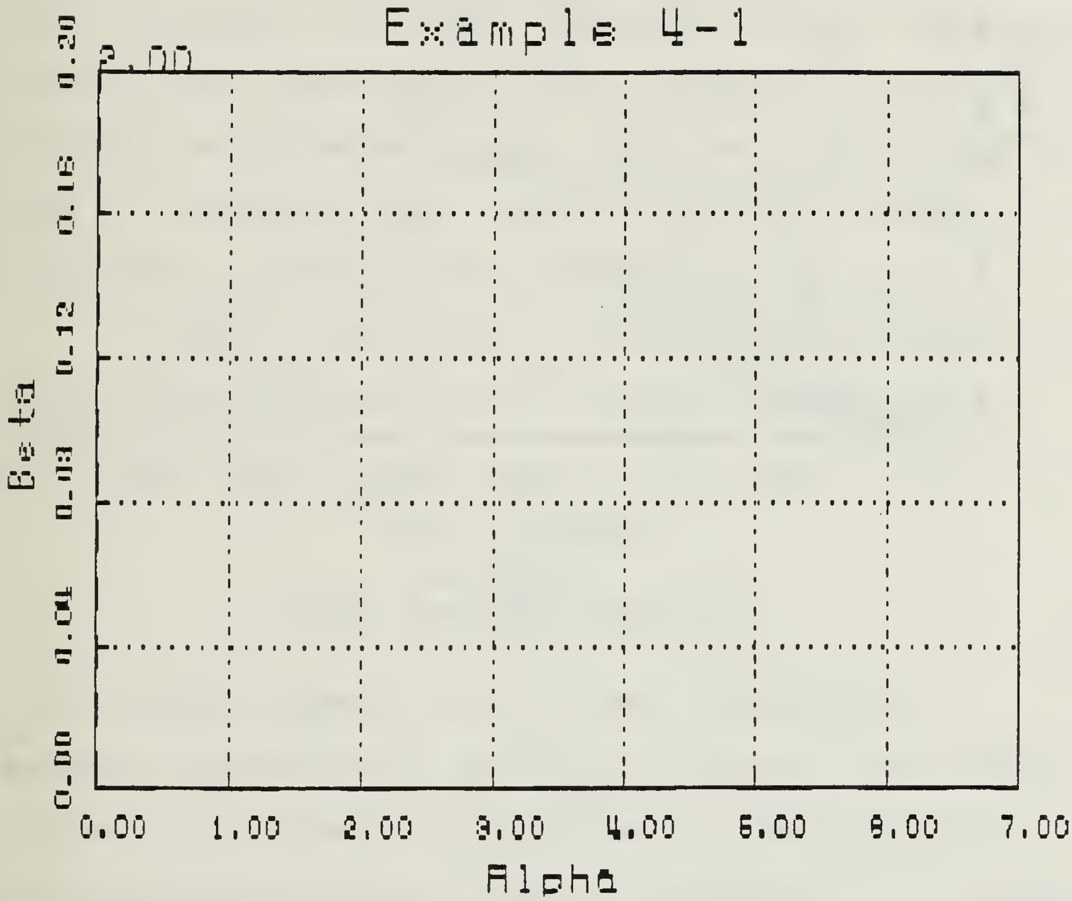

Figure 4-4
Constant Zeta-Omegan Curve

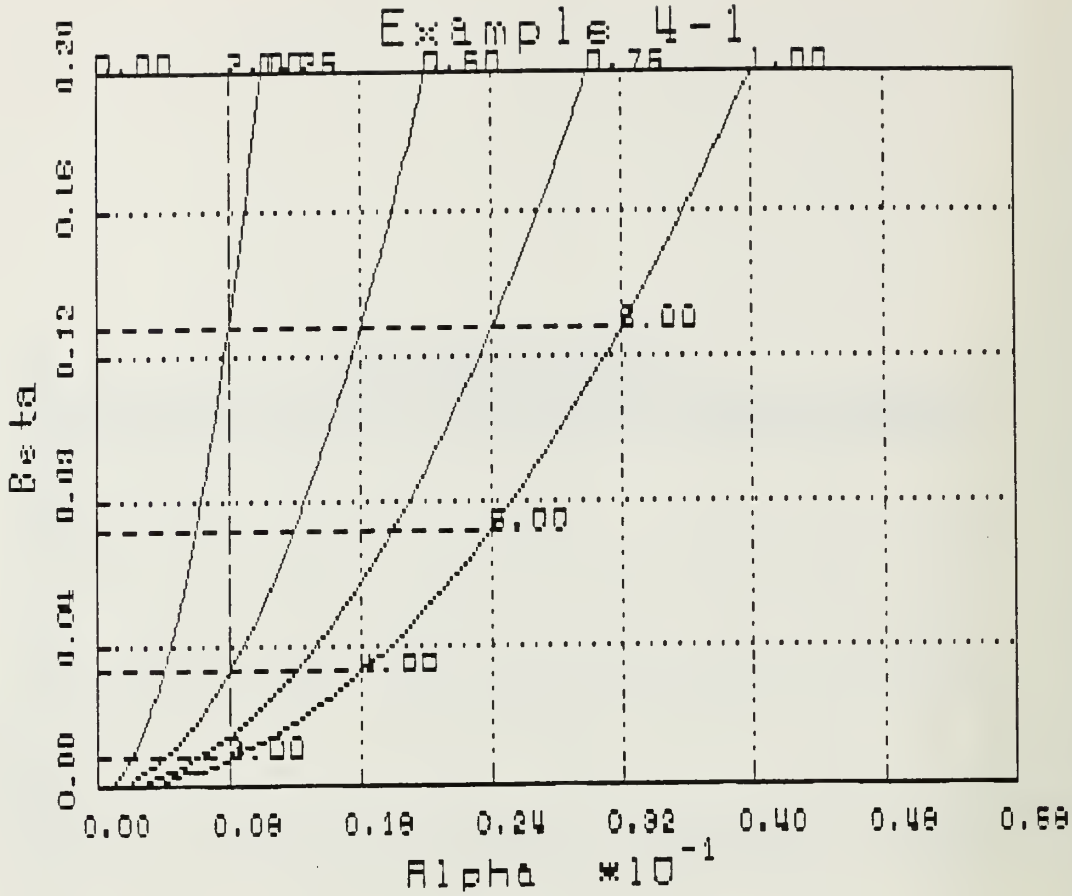

Figure 4-5
Constant Combined Curves

As expected due to the stable nature of this system, all curves are in the first quadrant because all coefficients must be positive to provide roots in the left half of the s-plane. A stable system requires compliance with this condition, although the condition alone does not guarantee stability.

At this point, we could select an option from the expansion menu to better determine the corresponding alpha and beta values satisfying our calculated zeta and omega

parameters. However, we can more precisely determine these values by plotting the actual desired zeta and omegan curves.

Exit from the Plotting Module. An option to save the contour values is offered but not necessary at this point. Reselect the CURVE Selection Option and enter the desired zeta and omegan curves. Equation (4-2) indicates that the zeta-omegan product must be greater than two to keep the settling time below two seconds. Therefore, also select for plotting a zeta-omegan curve of two.

Applying the previously discussed procedures for producing a plot provides the graph of Figure 4-6.

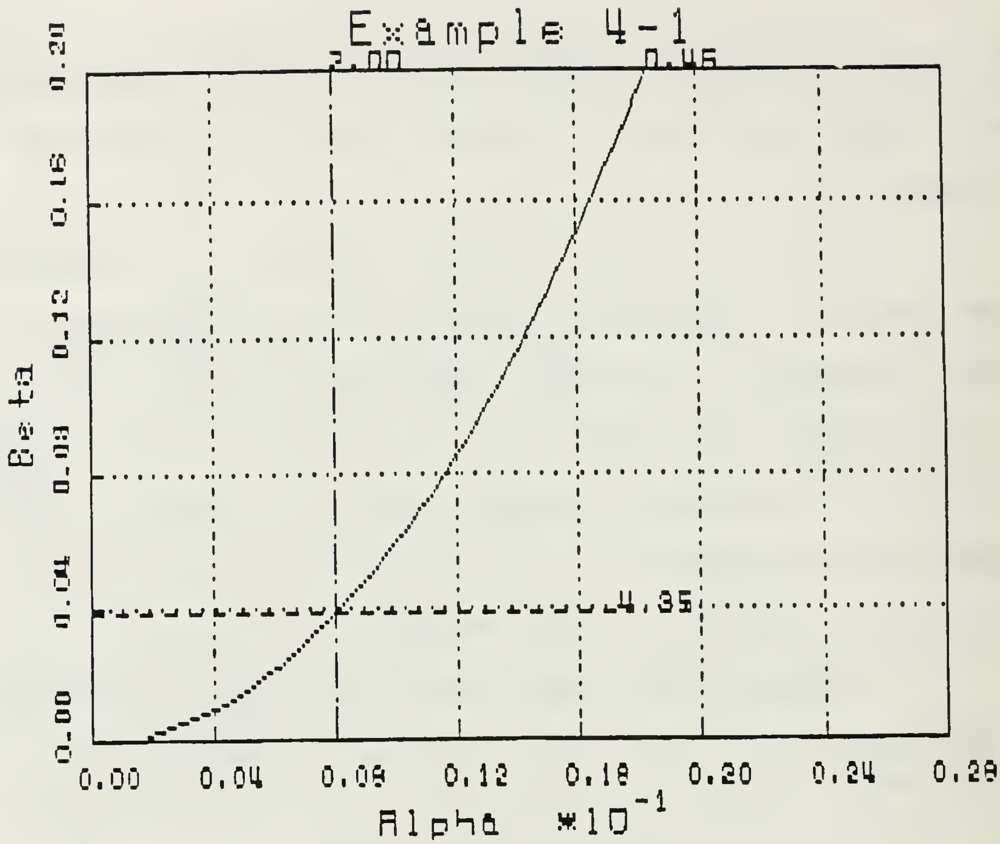

Figure 4-6
s**2 + 500As + 500B

All alpha/beta pairs falling to the left of the zeta-omegan curve satisfy the given settling time design requirement. The alpha/beta pair that just meets all requirements is the intersection of the three contours. This value can more accurately be determined by expanding around the position of this point. Using Option 1 of the expansion menu and inserting an alpha range from .006 to .010 and a beta range from .02 to .06 yields the following plot:

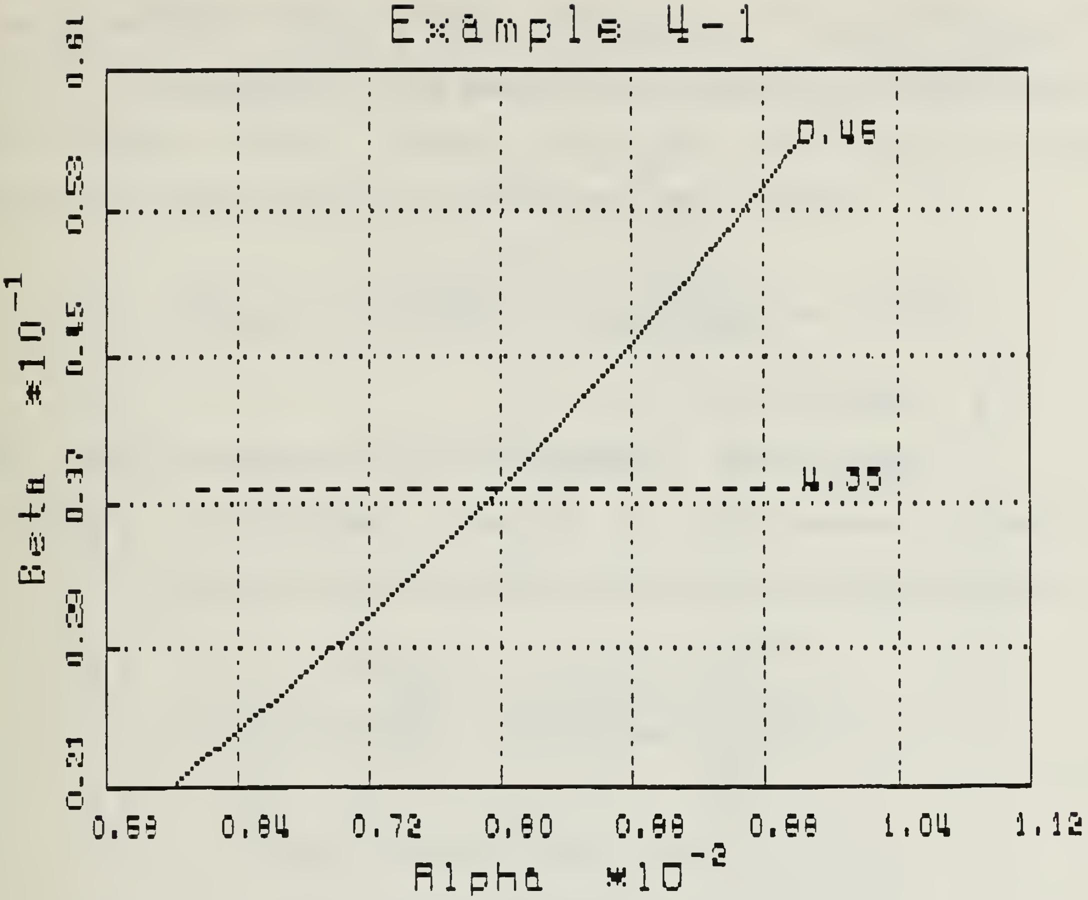

Figure 4-7
s**2 + 500As + 500B

The alpha and beta values determined from the preceding graph are:

$$\alpha = .008 \quad (4\text{-}13)$$

$$\beta = .038 \quad (4\text{-}14)$$

Therefore: K = .038 and K1 = .211

A determination of associated closed loop root values for this alpha/beta pair can be accomplished by selecting Option 8, ROOT Finder, in MAIN Menu. At the

prompt, enter the alpha and beta values. Confirming system stability, the outputted system root values are:

```
COEF( 3) =  1.00
COEF( 2) =  4.00
COEF( 1) =  19.0
First Root  =     -2.00    +   j 3.87
Second Root =     -2.00    -   j 3.87
```

2. Example 4-2

The system response of the previous plant could also be shaped with a cascade compensator. Figure 4-4 illustrates the block diagram of such a system.

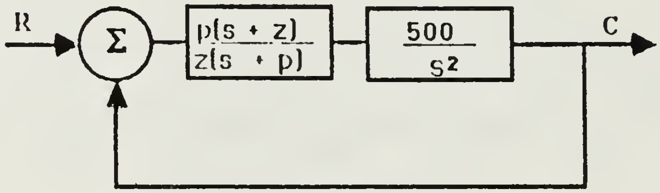

Figure 4-8
Ideal Instrument Servo with Cascade Compensator

The same procedures apply for deriving the parameter values associated with a particular desired system response.

Let:

$$\alpha = \frac{p}{z} \qquad (4\text{-}15)$$

$$\beta = p \qquad (4\text{-}16)$$

The characteristic equation for this system is then:

$$s^3 + \beta s^2 + 500\alpha s + 500\beta = 0 \qquad (4\text{-}17)$$

Since this is our first look at a third order system, a selection of zeta curves ranging from zero to one and omegan curves ranging from one to ten was again selected. The following graph was the result.

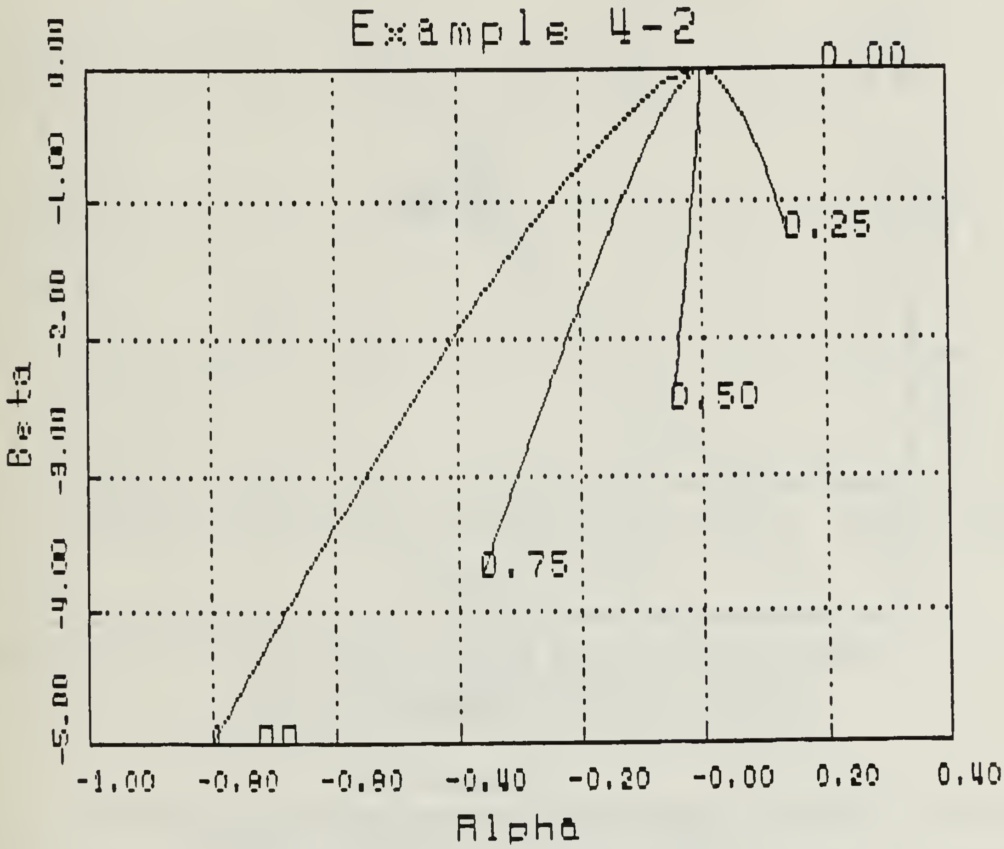

Figure 4-9
Constant Zeta Curves
s**3 + Bs**2 + 500As + 500B

As can be seen, most alpha/beta values are negative over the selected frequency range from 1 to 10 rad/s. Both parameters must be positive to provide all closed loop roots in the left half of the s-plane. Therefore, an alteration in frequency range is indicated. A range

spanning 10 to 40 rad/s was entered with the following results:

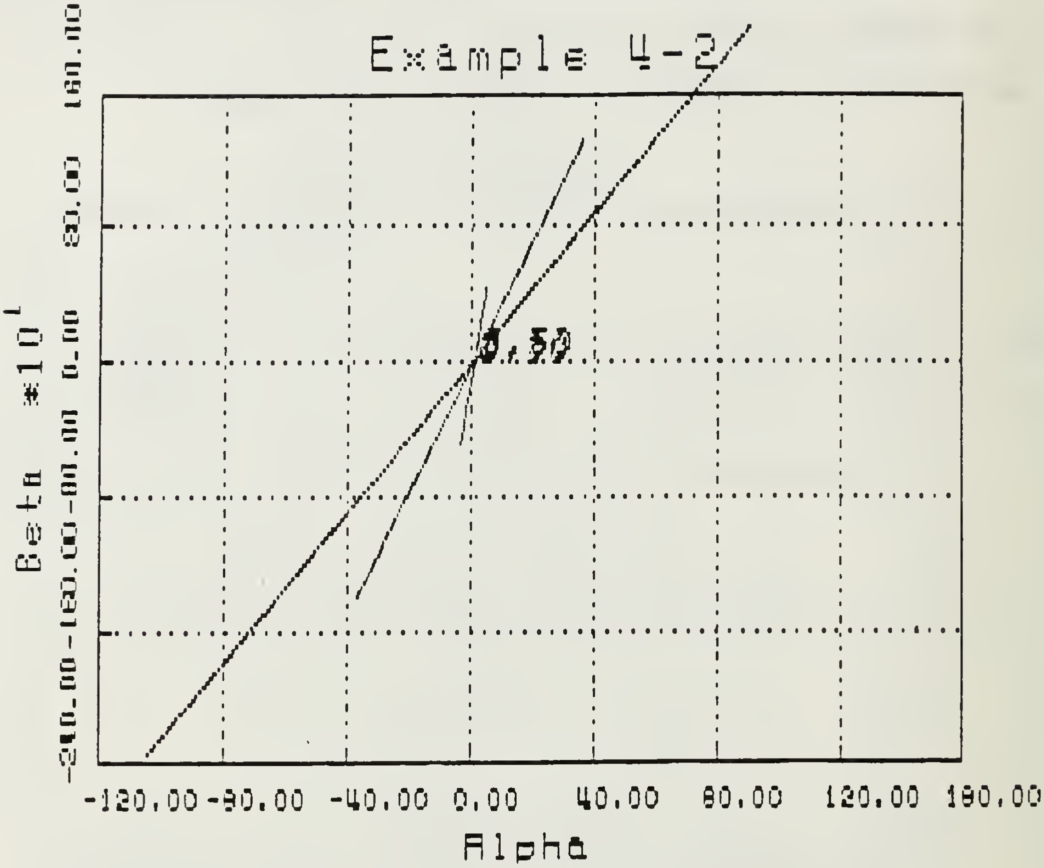

Figure 4-10
Constant Zeta Curves
s**3 + Bs**2 + 500As + 500B

It can now be seen that positive alpha/beta point pairs are possible, indicating that a stable system can be designed. However, specification limits cannot be met precisely due to the increased natural undamped frequency required. Figure 4-11 presents the zeta and omegan contours over a frequency range of 20 to 50 rad/s.

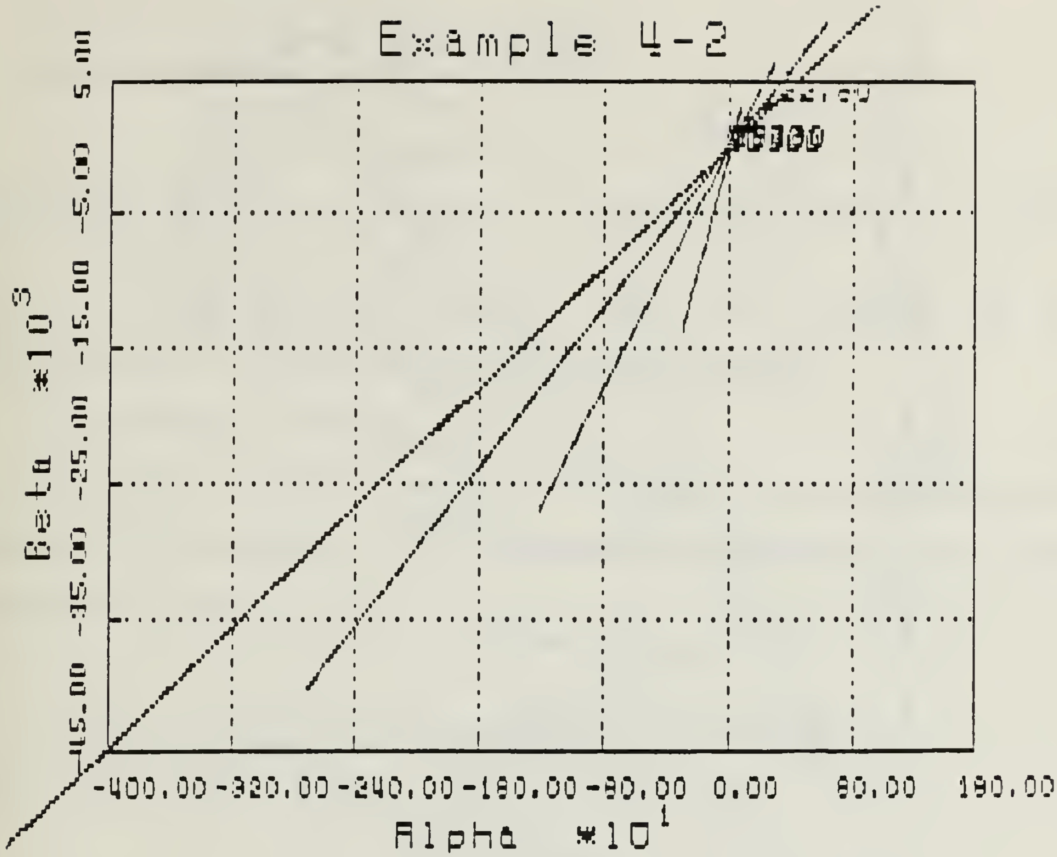

Figure 4-11
Constant Zeta & Omegan Curves
s**3 + Bs**2 + 500As + 500B

Expanding about an alpha range of zero to ten and beta range of zero to 200 yields a more easily interpreted graph as seen in Figure 4-12.

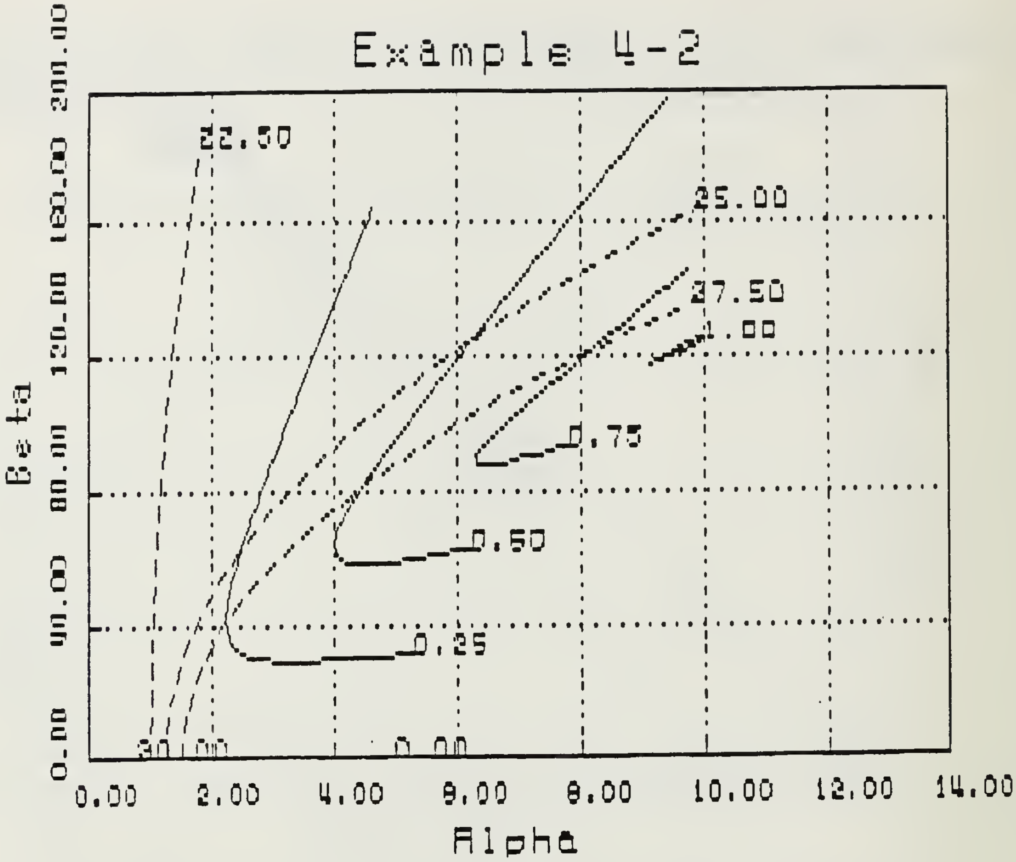

Figure 4-12
Constant Zeta & Omegan Curves
s**3 + Bs**2 + 500As + 500B

It appears that a minimum omegan value of 25 rad/s is required to maintain an acceptable relative damping coefficient. Selecting an operating point where zeta equals .5 and omegan equals 25 rad/s provides the alpha/beta pair:

$$\alpha = 6.3 \qquad (4\text{-}18)$$

$$\beta = 125 \qquad (4\text{-}19)$$

This corresponds to a compensator with a pole at 125 rad/s and a zero at 19.8 rad/s. The resulting closed loop roots with this set of parameters is:

```
                  COEF( 4) =   1.00

                  COEF( 3) =   125.

                  COEF( 2) =   .315E+04

                  COEF( 1) =   .625E+05

Root( 3) =  -12.65     + j  21.61          Root( 2) =  -12.65     - j  21.61
Root( 1) =  -99.69     + j  .0000
```

Thus, a clear set of dominant roots is present and all roots reside in the left half of the s-plane.

3. Example 4-3

This example deals with a plant incorporating both velocity (tachometer) and acceleration feedback. The block diagram of such a system is depicted in Figure 4-13.

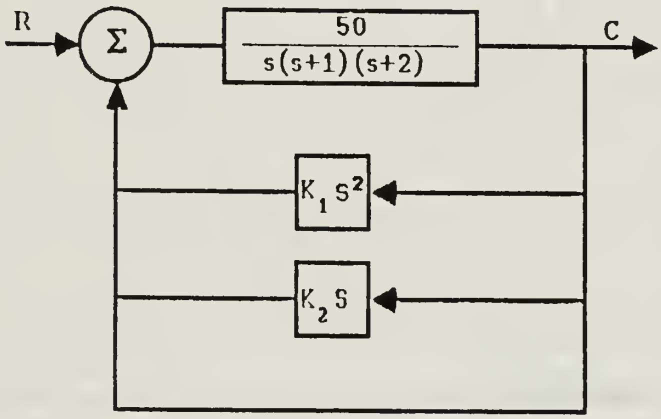

Figure 4-13
Velocity and Acceleration
Feedback Compensated System

The characteristic equation for this system is:

$$s^3 + (3+50\alpha)s^2 + (2+50\beta)s + 50 = 0 \qquad (4\text{-}20)$$

where:

$$\alpha = K_1 \qquad (4\text{-}21)$$

$$\beta = K_2 \qquad (4\text{-}22)$$

Rather than beginning with a required set of system responses, system options will first be examined to determine possible zeta and omegan values. Figure 4-14 provides a look at the combined zeta and omegan graph spanning a frequency range from one to 100 rad/s.

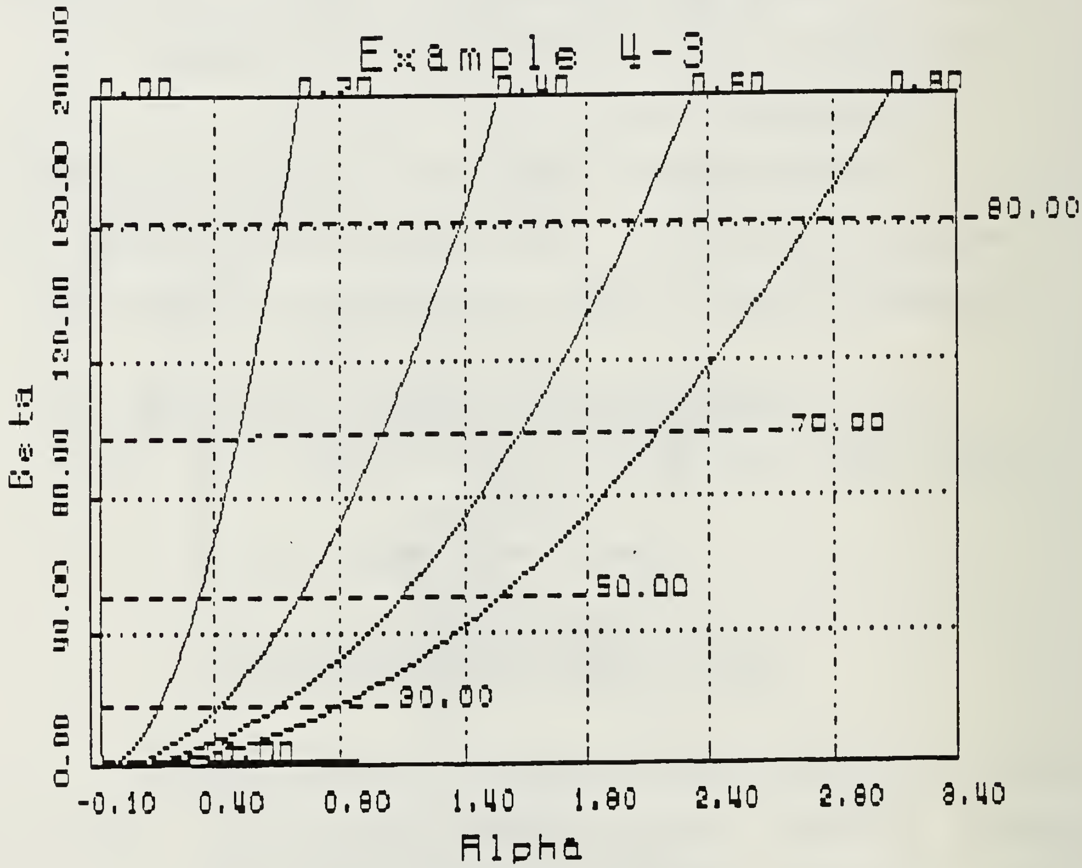

Figure 4-14
Constant Zeta & Omegan Curves
s**3 + (3+50A)s**2 + (2+50B)s + 50

This system appears to be stable over this frequency range. Design specifications requiring a settling time under one tenth of a second with an overshoot not to exceed ten percent translate into:

$$\zeta \geq .57 \tag{4-23}$$

$$\omega_n \geq 70 \ rad/s \tag{4-24}$$

Reentering the CURVE Selection Menu and adjusting the frequency range to 60-80 rad/s, while specifying curves of zeta equals .55, .57 and .60 and omegan equals 65, 70 and 75 rad/s, produces the plot of Figure 4-15.

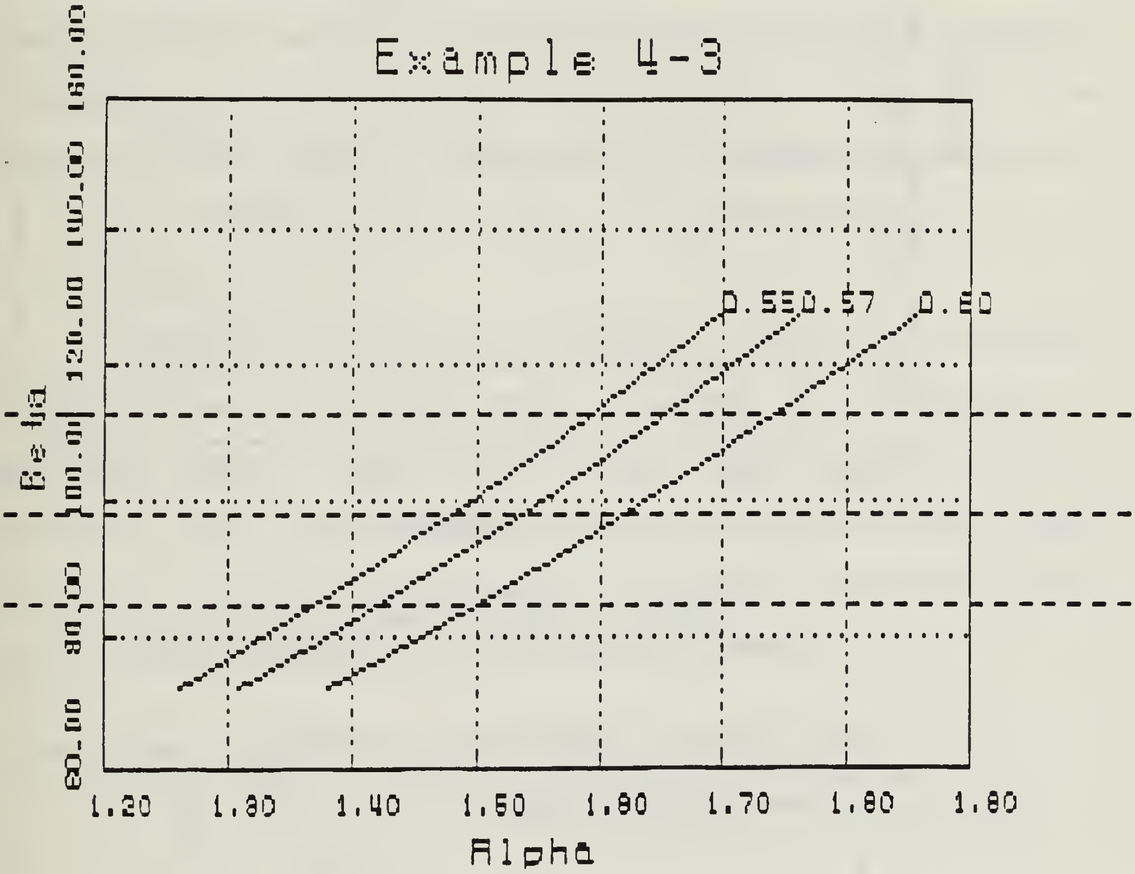

Figure 4-15
Constant Zeta & Omegan Curves
s**3 + (3+50A)s**2 + (2+50B)s + 50

Expanding about the desired operating point yields:

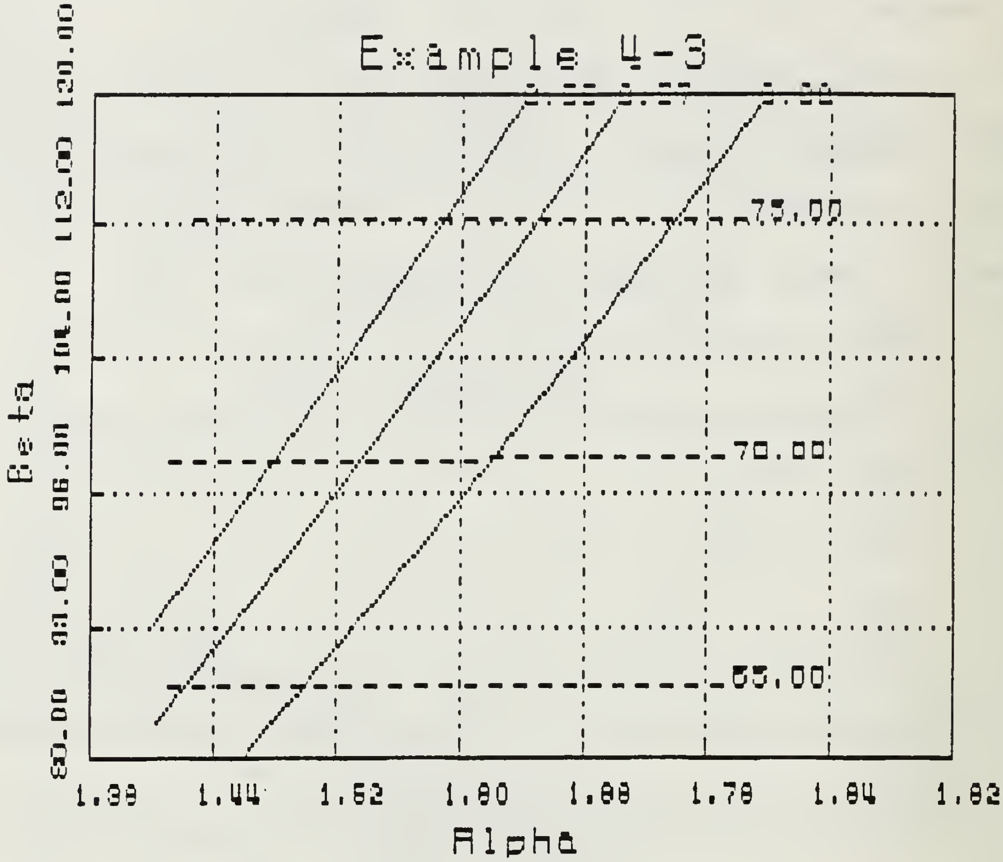

Figure 4-16
Constant Zeta & Omegan Curves
s**3 + (3+50A)s**2 + (2+50B)s + 50

From Figure 4-16, alpha and beta can be easily extracted. Their values are:

$$\alpha = 1.635 \tag{4-25}$$

$$\beta = 98.0 \tag{4-26}$$

Thus the acceleration feedback coefficient, K1, is 1.635 and the corresponding velocity feedback coefficient, K2, is 98.0. These values provide root locations of:

COEF(4) = 1.00

COEF(3) = 84.8

COEF(2) = .490E+04

COEF(1) = 50.0

```
Root( 3) =  -42.37      + j  55.73        Root( 2) =  -42.37      - j  55.73
Root( 1) =  -.1020E-01 + j  .0000
```

Obviously, the dominant roots are not at the designed locations. The dominant real root and relatively distant complex roots indicate that the system is probably overdamped. The omegan value chosen in an attempt to force a short settling time is too large. A redesign is indicated if a longer settling time can not be accommodated.

4. Example 4-4

The final example will examine a forth order system to demonstrate the difficulty involved in choosing an alpha/beta pair from a set of convoluted contours associated with higher order systems. A third order, type one plant will be shaped with a cascade compensator. The system block diagram is illustrated in Figure 4-17.

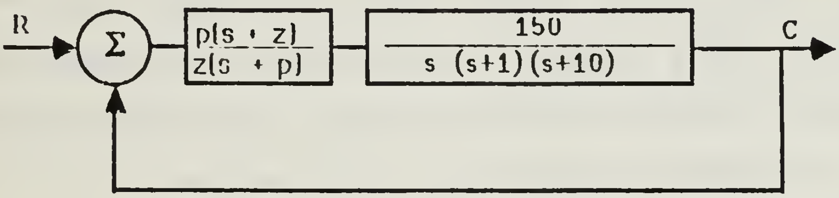

Figure 4-17
Third Order, Type One System
With Cascade Compensation

The characteristic equation for this system is:

$$s^4 + (11+\beta)s^3 + (10+11\beta)s^2 + (150\alpha+10\beta)s + 150\beta = 0 \qquad (4\text{-}27)$$

where:

$$\alpha = \frac{p}{z} \qquad (4\text{-}28)$$

$$\beta = p \qquad (4\text{-}29)$$

System specifications call for:

Peak Overshoot	< 50%
Settling Time	< 1 second

The corresponding relative damping coefficient and undamped natural frequency are:

$$\zeta \geq .22 \qquad (4\text{-}30)$$

$$\omega_n \geq 18.2 \; rad/s \qquad (4\text{-}31)$$

Based on Equation (4-7), the phase margin of a system with a zeta value of .22 is approximately 20 degrees. Although the system may be stable, it will be very sensitive, especially considering that the resonant peak of this system occurs at about the same frequency as omegan. In any case, with higher order systems it is important to perform a time response simulation to more accurately determine system behavior. Figure 4-18 is a combined plot over a frequency range of 10 to 50 rad/s.

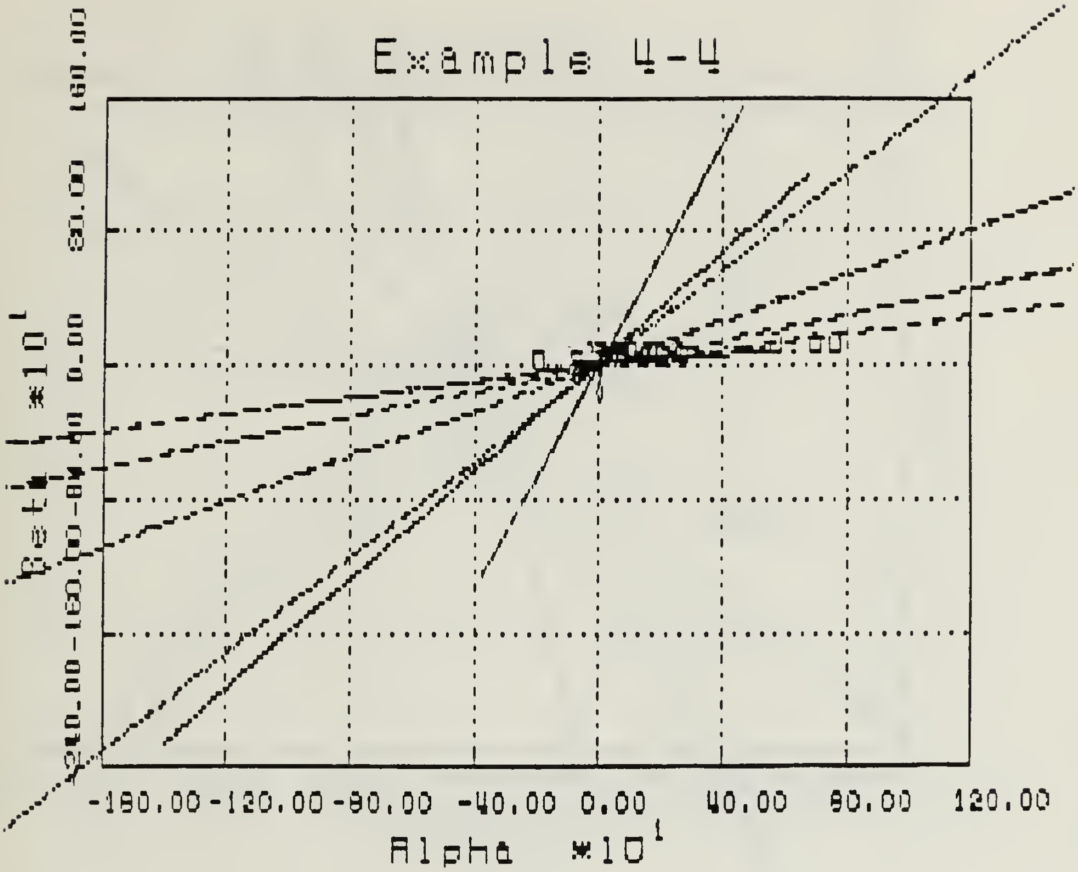

Figure 4-18
Constant Zeta & Omegan Curves
s**4 + (11+B)s**3 + (10+11B)s**2
+ (150A+10B)s + 150B

There appears to be a lot of activity about the operating point located at zero. Figure 4-19 is a result of expanding about that point.

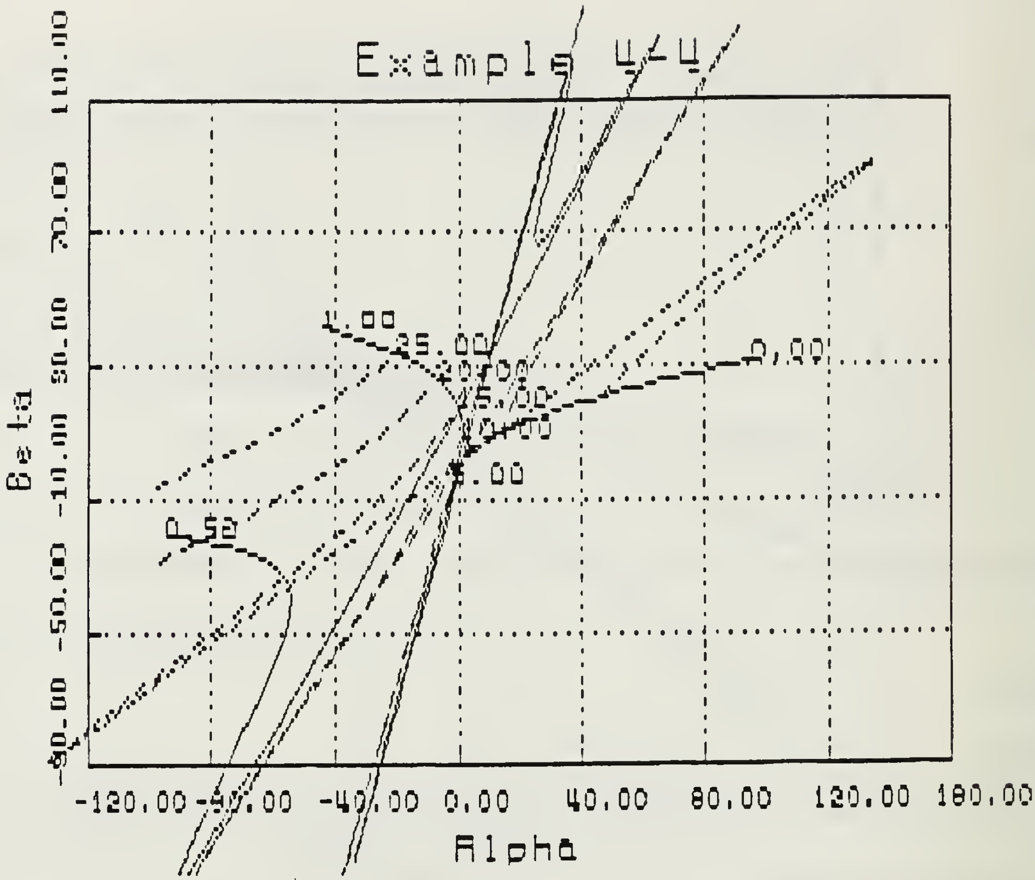

Figure 4-19
Constant Zeta & Omegan Curves
s**4 + (11+B)s**3 + (10+11B)s**2
+ (150A+10B)s + 150B

It is apparent that higher order plots are not as well behaved as the lower order systems. To reduce confusion and make the graph more readable, fewer contours must be selected. Reducing the range of frequencies to a span from 10 to 20 rad/s and selecting three zeta and two omega curves about the desired operating point produces the following graph.

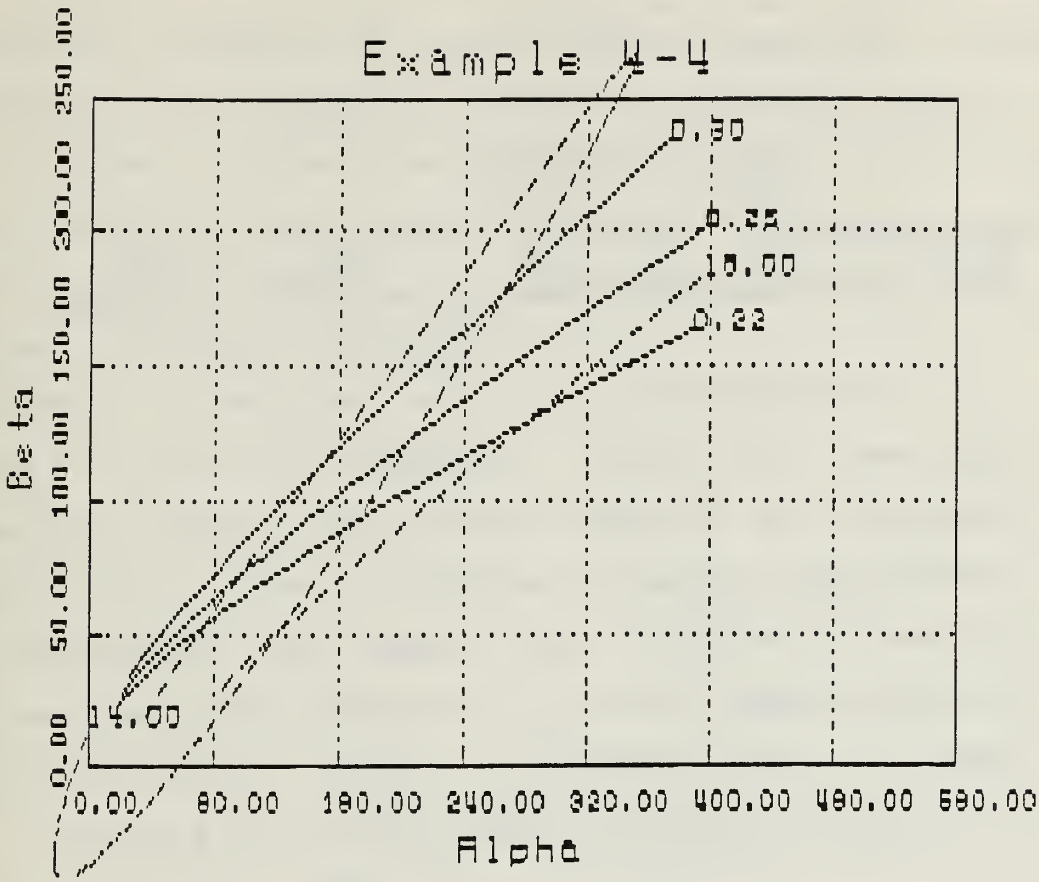

Figure 4-20
Constant Zeta & Omegan Curves
s**4 + (11+B)s**3 + (10+11B)s**2
+ (150A+10B)s + 150B

The alpha/beta pair associated with zeta equals .22 and omegan equals 18.0 follows:

$$\alpha = 280 \tag{4-32}$$

$$\beta = 130 \tag{4-33}$$

The roots derived through application of these values are:

```
COEF( 5) =  1.00

COEF( 4) =  141.

COEF( 3) =  .144E+04

COEF( 2) =  .433E+05

COEF( 1) =  .195E+05
Root( 4) =  -3.975     + j  17.49        Root( 3) =  -3.975     - j  17.49
Root( 2) =  -.4570     + j  .0000        Root( 1) =  -132.6     - j  .0000
Pause - Please enter a blank line (to continue) or a DOS command.
```

Unfortunately, with that particular selection of alpha and beta values, a dominant real root is present. Examination of the step input time response indicates an overshoot of 50% and a settling time of 1.2 seconds. An increase in zeta to better dampen the system results in a decreased omegan term if the system settling time is to remain at approximately one second. A zeta value increase to .3 with an associated omegan value of 14 yields:

$$\alpha = 150 \tag{4-34}$$

$$\beta = 115 \tag{4-35}$$

```
COEF( 5) =  1.00

COEF( 4) =  126.

COEF( 3) =  .128E+04

COEF( 2) =  .237E+05

COEF( 1) =  .173E+05
Root( 4) =  -4.218     + j  13.30        Root( 3) =  -4.218     - j  13.30
Root( 2) =  -.7591     + j  .0000        Root( 1) =  -116.7     - j  .0000
Pause - Please enter a blank line (to continue) or a DOS command.
```

The complex roots have made a relative shift in the s-plane toward the dominant real root. Once again, however, it can be difficult with high order systems to make a

definitive judgment on system behavior without viewing transient time response curves. The unit step response of this system provides a maximum overshoot of 40% with a one second settling time. Thus, a cascade compensator with a gain of 150, a zero at 0.76 and a pole at 115 provides a suitable transient response.

V. CONCLUSIONS AND RECOMMENDATIONS

A. CONCLUSIONS

As has been demonstrated, rapid solution of a wide range of linear automatic control systems problems can be achieved through use of the parameter plane program developed as part of this thesis. An IBM compatible microcomputer program offers the inherent advantages of portability and ready access to machines capable of supporting the code.

The graphical solution of the parameter plane technique provides the designer with a visual means of deducing how the dominant roots of the characteristic equation move about in the s-plane as two user defined parameters defining compensator attributes are varied. As a corollary, input of specific system response characteristics can be made to determine the parameter values providing that response. In addition, the parameter plane design procedure is independent of system configuration since it deals with the characteristic equation only.

There are some drawbacks to this technique. Only two variables can presently be considered in the computer model. However, a three variable parameter space method could be developed. In place of a series of curves in the two dimensional case, the parameter space graphical output

would be represented by a surface in three dimensions. Another shortcoming associated with the parameter plane analysis method is that information on open loop poles and zeros is not available.

B. RECOMMENDATIONS FOR FURTHER STUDY

The most significant addition that could be made to the parameter plane program would be the incorporation of a graphical cost function analysis package. In a linear feedback control system, the difference between the input to the system and its output is termed the error. This error differs with differing compensators. In most cases involving automatic feedback control systems, minimization of the error over a certain period of time (e.g., twice the settling time) is desired. Minimization of other system attributes is also common. One may wish to minimize the absolute value of the error, the square of the error or any one of an infinite number of cost functions, depending on the particular system application. Constant cost value contours can be plotted in much the same presentation as the parameter plane plot. Using standardized scales, desired system response characteristics can be compared with the error cost at that operating point and a suitable compromise between minimum cost and desired system response can be reached.

An alternative presentation method is available with the plotting package currently used by the parameter plane program. The cost value associated with a particular pair of alpha/beta values could be plotted as the third dimension of a parameter space surface. Although precise determination of the three variable values would be difficult, a quick synopsis of system response trends would be available. This would permit ready localization of interesting operating areas for more refined analysis on a two dimensional plot.

APPENDIX

PARAMETER PLANE PROGRAM

```
C
C ==========================================================================
C                         PARAMETER PLANE PROGRAM
C ==========================================================================
C
$NOfloatcalls
      Program ParaPlan
      dimension CONST(10),ALPHA(10),BETA(10),A(1002),B(1002),ZETA(10),
     *          WN(10),AOMEGA(1002),BOMEGA(1002),ZWN(10),AZTAOM(1002),
     *          BZTAOM(1002)
      real R,WNMIN,U,U1,U2,B1,B2,C1,C2,D1,D2,CONST,ALPHA,BETA,A,B,ZETA,
     *     WN,AOMEGA,BOMEGA,WNMAX,WNMIN1,ALPHA1,BETA1,ZWN,AZTAOM,
     *     BZTAOM,INCZWN
      integer SEL,SEL3,ORDER,NUMCOF,NUMZTA,I,M,N,K,L,J,ROW2,COLM,
     *        IOPORT,MODEL,START,STOP,FLAG,NUMWN,STOPO,FLAGRT,CURV,
     *        NUMZWN,STOPZO,OUT
      character*12 FILEIN,FILEOUT
      common A,B,AOMEGA,BOMEGA,AZTAOM,BZTAOM
      common/plottr/ IOPORT,MODEL
      data FLAGRT/0/
c
      IOPORT = 99
      MODEL = 99
      OUT = 3
      R = 10.
 10   call INTRMN (SEL)
      if ((SEL .eq. 4) .or. (SEL .eq. 6) .or. (SEL .eq. 10)) go to 4
 1    call MAINMN (SEL)
 4    FLAG = 0
 3    if (SEL .eq. 1) then
        call CRVSEL (CONST,ALPHA,BETA,ZETA,NUMZTA,STOP,NUMCOF,
     *               WNMIN,WNMAX,FLAG,SEL,NUMWN,WN,
     *               STOPO,ORDER,ZWN,NUMZWN,STOPZO)
      else if (SEL .eq. 2) then
        call MONPRT
      else if (SEL .eq. 3) then
        call REVCHG (ORDER,NUMCOF,CONST,ALPHA,BETA,WNMIN,WNMAX,SEL,FLAG)
      else if (SEL .eq. 4) then
        call LDFILE (ORDER,NUMCOF,CONST,ALPHA,BETA,WNMIN,WNMAX)
      else if (SEL .eq. 5) then
        call SAVFIL (ORDER,NUMCOF,CONST,ALPHA,BETA,WNMIN,WNMAX,FILEOUT)
      else if (SEL .eq. 6) then
        call CHAREQ (ORDER,CONST,ALPHA,BETA,NUMCOF,WNMIN,WNMAX,FLAGRT,
     *               ALPHA1,BETA1)
```

```
      else if (SEL .eq. 7) then
        call RUNPRO (NUMZTA,STOP,ZETA,CURV,NUMWN,WN,
     *                STOPO,ZWN,NUMZWN,STOPZO)
      else if (SEL .eq. 8) then
        FLAGRT = 1
        call ROOTS (ORDER,CONST,ALPHA,BETA,NUMCOF,WNMIN,WNMAX,
     *              ALPHA1,BETA1,FLAGRT,OUT,R)
        PAUSE
        FLAGRT = 0
      else if (SEL .eq. 10) then
        call SAMPLE
        goto 10
      else
        go to 999
 2    end if
      if (FLAG .eq. 1) go to 4
      go to 1
 999  stop
      end
      end
C
C
C **********************************************************************
C                          USER UTILITIES MODULE
C **********************************************************************
C
C ======================================================================
C  Subroutine  INTRMN -- allows input of C. E. from file or keyboard
C ======================================================================
C
      Subroutine INTRMN (SEL)
      integer SEL
 78   call CLR
      write(*,80)
      write(*,81)
      write(*,82)
      write(*,81)
      write(*,83)
      write(*,81)
      write(*,93)
 80   format(1x,///////,10x,'-----------------------------------------
     *---------------------',/,10x,'|',22x,'LOAD/INSERT MENU',24x,'|')
 81   format(10x,'----------------------------------------------------
     *----------')
 82   format(10x,'|',2x,'OPTION NO.',2x,'|',20x,'OPTION',21x,'|')
 83   format(10x,'|',2x,'    1     ',2x,'|',5x,'LOAD Problem from File',
```

```
     *20x,'|'
     *,/.10x,'|',2x,'     2      ',2x,'|',5x,'INPUT Characteristic Equatio
     *n',13x,'|'
     *,/,10x,'|',2x,'     3      ',2x,'|',5x,'EXAMPLE Characteristic Equat
     *ion Input',5x,'|'
     *,/,10x,'|',2x,'     9      ',2x,'|',5x,'EXIT to Main Menu',25x,'|')
 93   format(1x,///,15x,'Enter integer number for selection ===> ')
      call PSIT(22,56)
      read(*,*,err=78) SEL
      if (SEL .eq. 1) then
        SEL = 4
      else if (SEL .eq. 2) then
        SEL = 6
      else if (SEL .eq. 3) then
        SEL = 10
      else
        continue
      end if
      return
      end
      end
C
C ======================================================================
C  Subroutine  MAINMN -- allows selection of items from the main menu
C ======================================================================
C
      Subroutine MAINMN (SEL)
      integer SEL
 79   call CLR
      write(*,80)
      write(*,81)
      write(*,82)
      write(*,81)
      write(*,83)
      write(*,81)
      write(*,84)
      write(*,81)
      write(*,85)
      write(*,87)
      write(*,86)
      write(*,81)
      write(*,93)
 80   format(1x,//,10x,'-------------------------------------------------
     *----------------',/,10x,'|',25x,'MAIN MENU',28x,'|')
 81   format(10x,'-------------------------------------------------
     *----------')
```

```
87    format(10x,'++++++++++++++++++++++++++++++++++++++++++++++++++++++++
     *+++++++++++')
82    format(10x,'|',2x,'OPTION NO.',2x,'|',20x,'OPTION',21x,'|')
83    format(10x,'|',2x,'     1      ',2x,'|',5x,'CURVE Selection Menu',
     *22x,'|'
     *,/,10x,'|',2x,'     2      ',2x,'|',5x,'PRINTER Selection Menu'
     *,20x,'|'
     *,/,10x,'|',2x,'     3      ',2x,'|',5x,'REVIEW/CHANGE Selections'
     *,18x,'|')
84    format(10x,'|',2x,'     4      ',2x,'|',5x,'LOAD Problem from File',
     *20x,'|'
     *,/,10x,'|',2x,'     5      ',2x,'|',5x,'SAVE This Problem',
     *25x,'|'
     *,/,10x,'|',2x,'     6      ',2x,'|',5x,'INPUT Characteristic Equatio
     *n',13x,'|')
85    format(10x,'|',2x,'     7      ',2x,'|',5x,'PLOT Curves',31x,'|'
     *,/,10x,'|',2x,'     8      ',2x,'|',5x,'ROOT Finder',31x,'|')
86    format(10x,'|',2x,'     9      ',2x,'|',5x,'EXIT Program',30x,'|')
93    format(1x,///,15x,'Enter integer number for selection ===> ')
      call PSIT(24,56)
      read(*,*,err=79) SEL
      return
      end
C
C ======================================================================
C  Subroutine  LDFILE -- allows loading of data from an existing file
C ======================================================================
C
      Subroutine LDFILE (ORDER,NUMCOF,CONST,ALPHA,BETA,WNMIN,WNMAX)
      real WNMIN,WNMAX,CONST(10),ALPHA(10),BETA(10)
      integer ORDER,NUMCOF,MODEL,IOPORT,TEST,I
      character*12 FILEIN,ANS*1
      logical EXIST
      common/plottr/ IOPORT,MODEL
    1 call CLR
      write(*,100)
100   format(1x,////,10x,'-----------------------------------------------
     *------',/,10x,'*** This is the routine to LOAD data from a file **
     ***',/,10x,'----------------------------------------------------'
     *,//,10x,'File name should not exceed 8 characters',/,
     *10x,'File extension should not exceed 3 characters')
      write(*,101)
101   format(1x,////,10x,'What is the file name (fn) and extension (ext)
     *? ',//,10x,'Enter in form fn.ext (e.g. PARAPLAN.INP) ===> ')
      call PSIT(18,57)
      read(*,102) FILEIN
```

```
 102  format(a12)
      inquire(file=FILEIN,exist=EXIST)
      if (EXIST) then
        open(7,FILE=FILEIN,STATUS='OLD',ACCESS='SEQUENTIAL')
      else
        write(*,103)
 103    format(///,10x,'That file does not exist.',/,10x,
     *        'Do you want to try again?  (Y/N)')
        read(*,'(A1)') ANS
        if((ANS .eq. 'Y') .or. (ANS .eq. 'y')) then
          goto 1
        else
          goto 2
        end if
      end if
      read(7,105) MODEL,IOPORT
      read(7,110) ORDER,NUMCOF
      read(7,115) WNMIN,WNMAX
      do 190 I = 1,NUMCOF
        read(7,120) CONST(I),ALPHA(I),BETA(I)
190   continue
      close(7,STATUS='KEEP')
C
 105  format(1x,I2,2X,I2)
 110  format(1x,I2,2X,I2)
 115  format(1x,2F15.7)
 120  format(1x,3F15.7)
      write(*,3) WNMIN,WNMAX,ORDER
 3    format(1X,///////,5x,'WNMIN = ',f12.3,5x,'WNMAX = ',f12.3,5x,
     *        'ORDER = ',I2,//)
      do 9 I = NUMCOF,1,-1
        J = I - 1
      write(*,4) J,CONST(I),J,ALPHA(I),J,BETA(I)
 9    continue
 4    format(1X,'CONST(',I2,') = ',f10.3,5x,'ALPHA(',I2,') = ',f10.3,5x,
     *        'BETA(',I2,') =',f10.3)
      write(*,5)
 5    format(1X,///////)
      PAUSE
C
 2    return
      end
C
```

```
C ======================================================================
C  Subroutine  SAVFIL -- allows entered data to be saved to a data file
C ======================================================================
C
      Subroutine SAVFIL (ORDER,NUMCOF,CONST,ALPHA,BETA,WNMIN,WNMAX,
     *                   FILEOUT)
      real WNMIN,WNMAX,CONST(10),ALPHA(10),BETA(10)
      integer ORDER,NUMCOF,MODEL,IOPORT,TEST,I,J,SEL,SEL3
      character*12 FILEOUT,ANS*1
      logical EXIST
      common/plottr/ IOPORT,MODEL
   2  call CLR
      write(*,100)
 100  format(1x,////,10x,'-----------------------------------------------
     *----',/,10x,'*** This is the routine to SAVE data to a file ***'
     *,/,10x,'--------------------------------------------------'
     *,//,10x,'File name should not exceed 8 characters',/,
     *10x,'File extension should not exceed 3 characters')
      write(*,101)
 101  format(1x,////,10x,'What is the file name (fn) and extension (ext)
     *? ',//,10x,'Enter in form fn.ext (e.g. PARAPLAN.OUT) ===> ')
      call PSIT(18,57)
      read(*,102) FILEOUT
 102  format(a12)
      call CLR
      inquire(file=FILEOUT,exist=EXIST)
      if (EXIST) then
        write(*,103)
 103    format(///,10x,'That file already exists.',/,10x,
     *       'Do you want to overwrite?  (Y/N)')
        read(*,'(A1)') ANS
        if((ANS .eq. 'Y') .or. (ANS .eq. 'y')) then
          open(7,file=FILEOUT,status='OLD')
          rewind 7
          goto 1
        else
          goto 2
        end if
      end if
      open(7,file=FILEOUT,status='NEW')
   1  write(7,105) MODEL,IOPORT
      write(7,110) ORDER,NUMCOF
      write(7,115) WNMIN,WNMAX
      do 190 I = 1,NUMCOF
```

```
         write(7,120) CONST(I),ALPHA(I),BETA(I)
190    continue
       close(7,STATUS='KEEP')
C
 105   format(1x,I2,2X,I2)
 110   format(1x,I2,2X,I2)
 115   format(1x,2F15.7)
 120   format(1x,3F15.7)
       write(*,3) FILEOUT
 3     format(1x,////////,10x,'File name and extension = ',a12,///)
       write(*,4) WNMIN,WNMAX,ORDER
 4     format(5x,'WNMIN = ',g11.4,5x,'WNMAX = ',g11.4,5x,
     *        'ORDER = ',I2,//)
       do 9 I = NUMCOF,1,-1
         J = I - 1
         write(*,5) J,CONST(I),J,ALPHA(I),J,BETA(I)
 9     continue
 5     format(1X,'CONST(',I2,') = ',f10.3,5x,'ALPHA(',I2,') = ',f10.3,5x,
     *        'BETA(',I2,') =',f10.3)
       write(*,6)
 6     format(1x,//////)
       PAUSE
       return
       end
C
C ======================================================================
C  Subroutine  CHAREQ -- allows input of characteristic equation
C ======================================================================
C
       Subroutine CHAREQ (ORDER,CONST,ALPHA,BETA,NUMCOF,WNMIN,WNMAX,
     *                    FLAGRT,ALPHA1,BETA1)
       real CONST(10),ALPHA(10),BETA(10),WNMIN,WNMAX,ALPHA1,BETA1
       integer ORDER,I,NUMCOF,IMIN1,ROW1,FLAGRT
       character*1 CHG
 200   call CLR
C
       if (FLAGRT .eq. 1) then
       write(*,190)
 190   format(1x,//,9x,'Do you want to find the roots of the existing Cha
     *racteristic',/,9x,'Equation, entering values for ALPHA & BETA?',
     *//,9x,'Enter "y" or "n" ==> ')
       call PSIT(7,30)
       read(*,'(A)',err=200) CHG
       end if
       if ((CHG .eq. 'y') .or. (CHG .eq. 'Y')) then
         go to 293
```

```
      end if
      call CLR
C
C     *** Enter ORDER of Characteristic Equation ***
      write(*,201)
 201  format(1x,////,10x,'What is the order of the Characteristic Equati
     *on? ',//,10x,'Enter integer value less than or equal to 9 ===> ')
      call PSIT(8,59)
      read(*,*,err=200) ORDER
      NUMCOF = ORDER + 1
C
C     *** Enter CONSTANT Coefficients ***
 210  call CLR
      ROW1 = 10
      write(*,211)
 211  format(1x,////,5x,'Enter the CONSTANT Coefficient Values of the Ch
     *aracteristic Equation',/,5x,'-------------------------------------
     *------------------------------',/)
      do 217 I = NUMCOF,1,-1
        IMIN1 = I - 1
        write(*,220) IMIN1
 220    format(1x,/,5x,'CONSTANT Coefficient of S **',I2,' = ')
        call PSIT(ROW1,40)
        read(*,*,err=210) CONST(I)
        ROW1 = ROW1 + 2
 217  continue
 218  call CLR
      write(*,230)
 230  format(1x,////,20x,'***** CONSTANT COEFFICIENTS *****',//)
      do 235 I = NUMCOF,1,-1
        IMIN1 = I - 1
        write(*,237) IMIN1,CONST(I)
 237    format(26x,'CONST(',I2,') = ',f9.3,/)
 235  continue
      ROW1 = ROW1 + 3
      write(*,239)
 239  format(1x,//,5x,'Do you want to make any CHANGES to the CONSTANT C
     *oefficient Values?',//,5x,'Enter "y" or "n" ==> ')
      call PSIT(ROW1,26)
      read(*,'(A)',err=218) CHG
      if ((CHG .eq. 'y') .or. (CHG .eq. 'Y')) then
        go to 210
      end if
C
C     *** Enter ALPHA Coefficients ***
 240  call CLR
```

```
      ROW1 = 10
      write(*,241)
 241  format(1x,////,5x,'Enter the ALPHA Coefficient Values of the Chara
     *cteristic Equation',/,5x,'----------------------------------------
     *---------------------------',/)
      do 247 I = NUMCOF,1,-1
        IMIN1 = I - 1
        write(*,250) IMIN1
 250    format(1x,/,5x,'ALPHA Coefficient of S **',I2,' = ')
        call PSIT(ROW1,39)
        read(*,*,err=240) ALPHA(I)
        ROW1 = ROW1 + 2
 247  continue
C
      call CLR
      write(*,260)
 260  format(1x,////,20x,'***** ALPHA COEFFICIENTS *****',//)
      do 265 I = NUMCOF,1,-1
        IMIN1 = I - 1
        write(*,267) IMIN1,ALPHA(I)
 267    format(26x,'ALPHA(',I2,') = ',f9.3,/)
 265  continue
      ROW1 = ROW1 + 3
      write(*,269)
 269  format(1x,//,5x,'Do you want to make any CHANGES to the ALPHA Coef
     *ficient Values?',//,5x,'Enter "y" or "n" ==> ')
      call PSIT(ROW1,26)
      read(*,'(A)',err=240) CHG
      if ((CHG .eq. 'y') .or. (CHG .eq. 'Y')) then
        go to 240
      end if
C
C     *** Enter BETA Coefficients ***
 270  call CLR
      ROW1 = 10
      write(*,271)
 271  format(1x,////,5x,'Enter the BETA Coefficient Values of the Charac
     *teristic Equation',/,5x,'-----------------------------------------
     *-------------------------',/)
      do 277 I = NUMCOF,1,-1
        IMIN1 = I - 1
        write(*,280) IMIN1
 280    format(1x,/,5x,'BETA Coefficient of S **',I2,' = ')
        call PSIT(ROW1,38)
        read(*,*,err=270) BETA(I)
        ROW1 = ROW1 + 2
```

```
 277  continue
      call CLR
      write(*,290)
 290  format(1x,////,20x,'***** BETA COEFFICIENTS *****',//)
      do 292 I = NUMCOF,1,-1
        IMIN1 = I - 1
        write(*,294) IMIN1,BETA(I)
 294    format(26x,'BETA(',I2,') = ',f9.3,/)
 292  continue
      ROW1 = ROW1 + 3
      write(*,295)
 295  format(1x,//,5x,'Do you want to make any CHANGES to the BETA Coeff
     *icient Values?',//,5x,'Enter "y" or "n" ==> ')
      call PSIT(ROW1,26)
      read(*,'(A)',err=270) CHG
      if ((CHG .eq. 'y') .or. (CHG .eq. 'Y')) then
        go to 270
      end if
 293  call CLR
      CHG = 'n'
      if (FLAGRT .eq. 1) then
 370    write(*,371)
 371    format(/////,10x,'Enter ALPHA value  ==> ',/,10x,'All ALPHA coef
     *ficients multiplied by ALPHA')
        call PSIT(7,34)
        read(*,*,err=370) ALPHA1
 372    write(*,373)
 373    format(////,10x,'Enter BETA value  ==> ',/,10x,'All BETA coeffic
     *ients multiplied by BETA')
        call PSIT(12,33)
        read(*,*,err=372) BETA1
      end if
C
C     *** Enter RANGE of frequencies to examine ***
C
      if (FLAGRT .ne. 1) then
        write(*,297)
 297    format(1x,//,5x,'Enter the RANGE of frequencies you wish to exam
     *ine',/,5x,'--------------------------------------------------',//)
        write(*,298)
 298    format(1x,//,5x,'Enter minimum frequency (WnMIN) ===> ')
        call PSIT(10,43)
        read(*,*,err=293) WNMIN
 296    write(*,299)
 299    format(1x,//,5x,'Enter maximum frequency (WnMAX) ===> ')
        call PSIT(13,43)
```

```
        read(*,*,err=296) WNMAX
      end if
      return
      end
C
C ======================================================================
C  Subroutine  MONPRT -- allows display on monitor or printer
C ======================================================================
C
      Subroutine MONPRT
      integer IOPORT,MODEL,SEL
      common/plottr/ IOPORT,MODEL
C
      call CLR
      write(*,80)
      write(*,81)
      write(*,82)
      write(*,81)
      write(*,83)
      write(*,87)
      write(*,84)
      write(*,81)
      write(*,93)
 80   format(1x,///,10x,'-------------------------------------------------
     *-----------------',/,10x,'|',21x,'PRINTER/OUTPUT MENU',22x,'|')
 81   format(10x,'-------------------------------------------------------
     *-----------')
 87   format(10x,'++++++++++++++++++++++++++++++++++++++++++++++++++++++
     *++++++++++')
 82   format(10x,'|',2x,'PRINTER NO.',2x,'|',20x,'PRINTER',19x,'|')
 83   format(10x,'|',3x,'     1      ',2x,'|',4x,'Epson FX-80, All'
     *,26x,'|'
     *,/,10x,'|',3x,'     2      ',2x,'|',4x,'Epson FX-100, All'
     *,25x,'|'
     *,/,10x,'|',3x,'     3      ',2x,'|',4x,'Epson MX-100, All'
     *,25x,'|'
     *,/,10x,'|',3x,'     4      ',2x,'|',4x,'Epson RX-80, All'
     *,26x,'|'
     *,/,10x,'|',3x,'     5      ',2x,'|',4x,'Epson MX-80 & IBM Printer'
     *,17x,'|'
     *,/,10x,'|',3x,'     6      ',2x,'|',4x,'HP 7470A Graphics Plotter'
     *,17x,'|'
     *,/,10x,'|',3x,'     7      ',2x,'|',4x,'HP 7475A Graphics Plotter'
     *,17x,'|'
     *,/,10x,'|',3x,'     8      ',2x,'|',4x,'HP 758xB Series Plotters'
     *,18x,'|'
```

```
     *,/,10x,'|',3x,'    9      ',2x,'|',4x,'HP 2686A Laser Jet'
     *,24x,'|')
 84   format(10x,'|',3x,
     *'    10      ',2x,'|',4x,'Graphics Monitor (default)'
     *,16x,'|'
     *,/,10x,'|',3x,'   11      ',2x,'|',4x,'HARDWARE Interface Menu'
     *,19x,'|'
     *,/,10x,'|',3x,'   12      ',2x,'|',4x,'Input PLOT88 Values for IOPO
     *RT and MODEL'
     *,2x,'|'
     *,/,10x,'|',3x,'   99      ',2x,'|',4x,'EXIT to Main Menu',25x,'|')
 93   format(1x,///,15x,'Enter integer number for selection ===> ')
      call PSIT(24,56)
      read(*,*) SEL
      read(*,*) SEL
      if (SEL .eq. 1) then
        IOPORT = 0
        MODEL = 5
      else if (SEL .eq. 2) then
        IOPORT = 0
        MODEL = 15
      else if (SEL .eq. 3) then
        IOPORT = 0
        MODEL = 11
      else if (SEL .eq. 4) then
        IOPORT = 0
        MODEL = 1
      else if (SEL .eq. 5) then
        IOPORT = 0
        MODEL = 1
      else if (SEL .eq. 6) then
        IOPORT = 9600
        MODEL = 20
      else if (SEL .eq. 7) then
        IOPORT = 9600
        MODEL = 30
      else if (SEL .eq. 8) then
        IOPORT = 9600
        MODEL = 80
      else if (SEL .eq. 9) then
        IOPORT = 9600
        MODEL = 60
      else if (SEL .eq. 11) then
        call PORT
      else if (SEL .eq. 12) then
        call CLR
```

```
        write(*,100)

 100    format(1x,////,5x,'Input value to be used by the PLOT88 graphics
     * package',/,5x,'for IOPORT ==>',//)
        call PSIT(7,21)
        read(*,*) IOPORT
        write(*,110)
 110    format(1x,///,5x,'Input value to be used by the PLOT88 graphics
     *package',/,5x,'for MODEL ==>',//)
        call PSIT(12,20)
        read(*,*) MODEL
      else
        IOPORT = 99
        MODEL = 99
      endif
      write(*,98) IOPORT,MODEL
 98   format(//,10x,'PLOT88 will use these values to output graphics:'
     *,//,10x,'IOPORT = ',I4,10x,'MODEL = ',I4,///)
      PAUSE
      return
      end
C
C ======================================================================
C  Subroutine  PORT -- allows selection of output printer port
C ======================================================================
C
      Subroutine PORT
      integer IOPORT,MODEL,SEL
      common/plottr/ IOPORT,MODEL
C
      call CLR
      write(*,79)
 79   format(1x,///,15x,'Selection of a printer in the previous menu aut
     *omatically',/,10x,'selects the most commonly associated output por
     *t, be it parallel ',/,10x,
     *'(LPT) or serial, for that particular device. If the program graph
     *ics',/,10x,
     *'are not being output correctly, use this menu to properly route t
     *he',/,10x,
     *'graphics data to the output device.',////)
      PAUSE
C
      call CLR
      write(*,80)
      write(*,81)
      write(*,82)
```

```
      write(*,81)
      write(*,83)
      write(*,87)
      write(*,84)
      write(*,81)
      write(*,93)
 80   format(1x,///,10x,'--------------------------------------------------
     *-----------------',/,10x,'|',19x,'HARDWARE INTERFACE MENU'
     *,20x,'|')
 81   format(10x,'--------------------------------------------------
     *-----------')
 87   format(10x,'++++++++++++++++++++++++++++++++++++++++++++++++++
     *++++++++++')
 82   format(10x,'|',2x,'SELECT NO.',3x,'|',20x,'PORT',22x,'|')
 83   format(10x,'|',3x,'    1      ',2x,'|',14x,'LPT1 Printer Port'
     *,15x,'|'
     *,/,10x,'|',3x,'    2      ',2x,'|',14x,'LPT2 Printer Port'
     *,15x,'|'
     *,/,10x,'|',3x,'    3      ',2x,'|',14x,'LPT3 Printer Port'
     *,15x,'|'
     *,/,10x.'|',3x,'    4      ',2x,'|',14x,'COM1 Serial Port'
     *,16x,'|'
     *,/,10x,'|',3x,'    5      ',2x,'|',14x,'COM2 Serial Port'
     *,16x,'|')
 84   format(10x,'|',3x,
     *'    99     ',2x,'|',14x,'EXIT to Main Menu',15x,'|')
 93   format(1x,///,15x,'Enter integer number for selection ===> ')
      call PSIT(21,56)
      read(*,*) SEL
      if (SEL .eq. 1) then
        IOPORT = 0
      else if (SEL .eq. 2) then
        IOPORT = 2
      else if (SEL .eq. 3) then
        IOPORT = 3
      else if (SEL .eq. 4) then
        IOPORT = 0
      else if (SEL .eq. 5) then
        IOPORT = 50
      else
        IOPORT = 99
      endif
      if ((SEL .eq. 4) .or. (SEL .eq. 5)) then
      if ((SEL .eq. 4) .or. (SEL .eq. 5)) then
C
        call CLR
```

```
      write(*,60)
      write(*,61)
      write(*,62)
      write(*,61)
      write(*,63)
      write(*,61)
      write(*,93)
60   format(1x,///,10x,'------------------------------------------
    *-----------------',/,10x,'|',16x,'BAUD (data transfer) RATE MENU'
    *,16x,'|')
61   format(10x,'-----------------------------------------------------
    *----------')
62   format(10x,'|',2x,'SELECT NO.',3x,'|',15x,'BAUD RATE',22x,'|')
63   format(10x,'|',3x,'     1      ',2x,'|',18x,' 300'
    *,24x,'|'
    *,/,10x,'|',3x,'     2      ',2x,'|',18x,'1200'
    *,24x,'|'
    *,/,10x,'|',3x,'     3      ',2x,'|',18x,'4800'
    *,24x,'|'
    *,/,10x,'|',3x,'     4      ',2x,'|',18x,'9600'
    *,24x,'|')
     call PSIT(18,56)
     read(*,*) SEL
     if (SEL .eq. 1) then
       IOPORT = IOPORT + 300
     else if (SEL .eq. 2) then
       IOPORT = IOPORT + 1200
     else if (SEL .eq. 3) then
       IOPORT = IOPORT + 4800
     else
       IOPORT = IOPORT + 9600
     endif
C
       call CLR
       write(*,50)
       write(*,51)
       write(*,52)
       write(*,51)
       write(*,53)
       write(*,51)
       write(*,93)
50   format(1x,///,10x,'------------------------------------------
    *-----------------',/,10x,'|',25x,'PARITY MENU'
    *,26x,'|')
51   format(10x,'-----------------------------------------------------
    *----------')
```

```
52    format(10x,'|',2x,'SELECT NO.',3x,'|',17x,'PARITY',23x,'|')

53    format(10x,'|',3x,'     1      ',2x,'|',14x,' NO Parity'
     *,22x,'|'
     *,/,10x,'|',3x,'     2      ',2x,'|',14x,'EVEN Parity'
     *,21x,'|'
     *,/,10x,'|',3x,'     3      ',2x,'|',14x,'ODD Parity'
     *,22x,'|')
      call PSIT(17,56)
      read(*,*) SEL
      end if
      if (SEL .eq. 3) then
        IOPORT = IOPORT + 1
      else if (SEL .eq. 2) then
        IOPORT = IOPORT + 2
      endif
 99   return
      end
C
C ========================================================================
C  Subroutine  SAMPLE -- demonstrates input of characteristic equation
C ========================================================================
C
      Subroutine SAMPLE
C
      call CLR
      write(*,10)
 10   format(1x,////,1x,'---------------------------------------------
     *------------------------------',/,5x,
     *'*** This subroutine demonstrates loading a Characteristic Equatio
     *n ***',/,
     *1x,'-------------------------------------------------------------
     *---------------',//)
      write(*,20)
 20   format(1x,
     *'A universal third order system is represented by the characterist
     *ic equation:',//,20x,'s**3 + As**2 + Bs + 1 = 0',//,1x,
     *'A and B are variables representing user defined compensator param
     *eters.',//,10x,
     *'The characteristic equation is obtained with a 1/s**3 plant',/,
     *5x,'incorporating both acceleration (s**2) and velocity (s)',/
     *,5x,'feedback and assigning variables A and B as the respective',/
     *,5x,'gain terms of the two compensators.',//,10x,
     *'To load this characteristic equation into the computer model firs
     *t',/,5x,
     *'select option 2 in the introductory LOAD/INSERT Menu or option 6
```

```
     *in the',/,5x,

     *'MAIN Menu. Both options are titled "INPUT Characteristic Equation
     *".',//)
      PAUSE
      call CLR
      write(*,30)
30    format(1x,//,10x,
     *'You will be asked to insert CONSTANT, ALPHA and BETA coefficients
     *.',/,5x,
     *'In the case of the universal third order system, the appropriate
     *response',/,5x,'following each prompt is:',//,20x,
     *'CONSTANT Coefficient of S ** 3 =   1',/,20x,
     *'CONSTANT Coefficient of S ** 2 =   0',/,20x,
     *'CONSTANT Coefficient of S ** 1 =   0',/,20x,
     *'CONSTANT Coefficient of S ** 0 =   1',//,20x,
     *'ALPHA Coefficient of S ** 3 =      0',/,20x,
     *'ALPHA Coefficient of S ** 2 =      1',/,20x,
     *'ALPHA Coefficient of S ** 1 =      0',/,20x,
     *'ALPHA Coefficient of S ** 0 =      0',//,20x,
     *'BETA Coefficient of S ** 3 =       0',/,20x,
     *'BETA Coefficient of S ** 2 =       0',/,20x,
     *'BETA Coefficient of S ** 1 =       1',/,20x,
     *'BETA Coefficient of S ** 0 =       0',/,20x,////)
      PAUSE
      call CLR
      write(*,40)
40    format(1x,//,5x,
     *'When each set of responses is complete, they will be echoed back
     *for',/,5x,
     *'verification.',//,10x,
     *'You will next be asked to enter the minimum and maximum frequenci
     *es',/,5x,
     *'you wish to examine. This is the natural undamped frequency range
     * over',/,5x,
     *'which the relative damping coefficient (zeta) will be calculated
     *to',/,5x,
     *'construct the constant zeta curves.',//,10x,
     *'Generally, the minimum and maximum frequencies will be dictated b
     *y the',/,5x,
     *'desired system response. For example, we wish to have a system se
     *ttling',/,5x,
     *'time of less than 2 seconds. Settling time is approximated by the
     * equation',/,5x,
     *'4 divided by the product of relative damping coefficient and unda
     *mped',/,5x,
```

```
     *'natural frequency. If we wish to look at a range of zeta values f
     *rom .1',/,5x,
     *'to 1, we can solve for the applicable min and max frequencies. In
     * this',/,5x,
     *'case they are 2 Hz and 20 Hz.',////)
      PAUSE

      return
      end
C
C ======================================================================
C  Subroutine  REVCHG -- allows review or change of entered data
C ======================================================================
C
      Subroutine REVCHG (ORDER,NUMCOF,CONST,ALPHA,BETA,WNMIN,WNMAX,
     *                   SEL,FLAG)
      real CONST(10),ALPHA(10),BETA(10),WNMIN,WNMAX
      integer CHG,ORDER,NUMCOF,FLAG,SEL,ROW
      call CLR
 90   write(*,100)
 100  format(1x,///,10x,'***** Characteristic Equation Coefficients ****
     ***',///)
      write(*,3) WNMIN,WNMAX,ORDER
 3    format(5x,'WNMIN = ',f12.3,5x,'WNMAX = ',f12.3,5x,
     *       'ORDER = ',I2,//)
      do 9 I = NUMCOF,1,-1
        J = I - 1
      write(*,4) J,CONST(I),J,ALPHA(I),J,BETA(I)
 9    continue
 4    format(1X,'CONST(',I2,') = ',f10.3,5x,'ALPHA(',I2,') = ',f10.3,5x,
     *       'BETA(',I2,') =',f10.3)
      write(*,5)
 5    format(1x,//,5x,'Do you want to make any CHANGES to the Coefficien
     *t Values?',//,5x,'Enter "y" or "n" ==> ')
      ROW = 16 + NUMCOF
      call PSIT(ROW,27)
      read(*,'(A)',err=90) CHG
      if ((CHG .eq. 'y') .or. (CHG .eq. 'Y')) then
        SEL = 6
        FLAG = 1
      end if
      return
      end
```

```
C
C ***********************************************************************
C                          CURVE SELECTION MODULE
C ***********************************************************************
C
C
C =======================================================================
C  Subroutine  CRVSEL -- allows selection of desired constant curves
C =======================================================================
C
      Subroutine CRVSEL (CONST,ALPHA,BETA,ZETA,NUMZTA,STOP,NUMCOF,
     *                   WNMIN,WNMAX,FLAG,SEL,NUMWN,WN,
     *                   STOPO,ORDER,ZWN,NUMZWN,STOPZO)
      real R,WNMIN,U,U1,U2,B1,B2,C1,C2,D1,D2,ZETA(10),INCZOM,
     *     WNMAX,WNMIN1,CONST(10),ALPHA(10),BETA(10),A(1002),B(1002),
     *     INCLOG,MAXLOG,MINLOG,WN(10),INC1,ZETA1,AOMEGA(1002),INCZWN,
     *     BOMEGA(1002),ALPHA1,BETA1,ZWN(10),AZTAOM(1002),BZTAOM(1002)
      integer SEL,SEL3,ORDER,NUMCOF,NUMZTA,I,M,N,K,L,ROW2,COLM,
     *        START,STOP,FLAG,FRQCHG,NUMWN,TABLE,
     *        STARTO,STOPO,ZTASEL,FLAGRT,NUMZWN,STOPZO,OUT
      character*1 ANS
      character*12 FILENAME
      common A,B,AOMEGA,BOMEGA,AZTAOM,BZTAOM
      common/filenam/ FILENAME
      logical EXIST
      FILENAME = '  '
C
 20     call CLR
        write(*,21)
        write(*,32)
        write(*,22)
        write(*,32)
        write(*,23)
        write(*,32)
        write(*,38)
 21   format(1x,//,10x,'----------------------------------------------
     *----------------',/,10x,'|',16x,'CURVE DATA POINT DISPLAY MENU',
     *17x,'|')
 22   format(10x,'|',2x,'OPTION NO.',2x,'|',15x,'OUTPUT SELECTION'
     *,16x,'|')
 23   format(10x,'|',2x,'     1     ',2x,'|',5x,'NO output DISPLAY or sav
     *e to FILE',9x,'|'
     *,/,10x,'|',2x,'     2     ',2x,'|',5x,'DISPLAY every 10th alpha/bet
     *a value',7x,'|'
     *,/,10x,'|',2x,'     3     ',2x,'|',5x,'DISPLAY alpha/beta values an
     *d roots',7x,'|'
```

```
    *,/,10x,'|',2x,'    4      ',2x,'|',5x,'DISPLAY & FILE alpha/betas a
    *nd roots',6x,'|'
    *,/,10x,'|',2x,'    5      ',2x,'|',5x,'FILE all alpha/beta values'
    *,/,10x,'|',2x,'    6      ',2x,'|',5x,'FILE all alpha/beta values a
    *nd roots',6x,'|')
28   format(1x,///,15x,'Enter integer number for selection ===> ')
     call PSIT(19,56)
     read(*,*,err=20) OUT
     call CLR
     if ((OUT .lt. 4) .or. (OUT .gt. 6)) go to 30
 29  write(*,101)
101  format(1x,////,10x,'What is the file name (fn) and extension (ext)
    *? ',//,10x,'Enter in form fn.ext (.ext is optional) ===> ')
     call PSIT(8,56)
     read(*,102) FILENAME
102  format(a12)
     inquire(file=FILENAME,exist=EXIST)
     if (EXIST) then
       write(*,103)
103    format(///,10x,'That file already exists.',/,10x,
    *          'Do you want to overwrite?  (Y/N)')
       read(*,'(A1)') ANS
       if((ANS .eq. 'Y') .or. (ANS .eq. 'y')) then
         open(8,file=FILENAME,status='OLD')
         rewind 8
         goto 30
       else
         goto 29
       end if
     end if
1    open(8,file=FILENAME,status='NEW')
     write(8,196) WNMIN,WNMAX
196  format(1x,///,10x,'WNMIN =',g11.4,9x,'WNMAX ='
    *                 ,g11.4,//)
30     call CLR
       write(*,31)
       write(*,32)
       write(*,33)
       write(*,32)
       write(*,34)
       write(*,37)
       write(*,35)
       write(*,37)
       write(*,36)
       write(*,32)
       write(*,38)
```

```
 31   format(1x,//,10x,'--------------------------------------------------
     *----------------',/,10x,'|',19x,'CURVE SELECTION MENU',23x,'|')
 32   format(10x,'--------------------------------------------------------
     *----------')
 33   format(10x,'|',2x,'OPTION NO.',2x,'|',16x,'CURVE SELECTION'
     *,16x,'|')
 34   format(10x,'|',2x,'     1      ',2x,'|',5x,'Constant ZETA Curves'
     *,22x,'|'
     *,/,10x,'|',2x,'     2      ',2x,'|',5x,'Constant OMEGA Curves'
     *,21x,'|'
     *,/,10x,'|',2x,'     3      ',2x,'|',5x,'Constant ZETA*OMEGA Curves'
     *,16x,'|'
     *,/,10x,'|',2x,'     4      ',2x,'|',5x,'Change Frequency Range'
     *,20x,'|')
 35   format(10x,'|',2x,'     5      ',2x,'|',5x,'PLOT Selected Curves'
     *,22x,'|')
 36   format(10x,'|',2x,'     9      ',2x,'|',5x,'EXIT to Main Menu'
     *,25x,'|')
 37   format(10x,'++++++++++++++++++++++++++++++++++++++++++++++++++++++
     *++++++++++')
 38   format(1x,///,15x,'Enter integer number for selection ===> ')
      call PSIT(21,56)
      read(*,*,err=30) SEL3
      FLAG = 0
      call CLR
C
C
C                            *** CONSTANT ZETA PLOTS ***
C
 330  if (SEL3 .eq. 1) then
        if (FRQCHG .eq. 1) go to 380
C
C . . . Constant ZETA curve selection menu
C
 40   call CLR
      write(*,41)
      write(*,42)
      write(*,43)
      write(*,42)
      write(*,44)
      write(*,46)
      write(*,45)
      write(*,42)
      write(*,47)
 41   format(1x,//,10x,'--------------------------------------------------
     *----------------',/,10x,'|',19x,'CONSTANT ZETA MENU',25x,'|')
```

```
 42   format(10x,'-------------------------------------------------------
     *----------')
 43   format(10x,'|',2x,'OPTION NO.',2x,'|',14x,'ZETA CURVE SELECTION'
     *,13x,'|')
 44   format(10x,'|',2x,'    1     ',2x,'|',5x,'Select particular consta
     *nt ZETA curves',4x,'|'
     *,/,10x,'|',2x,'    2     ',2x,'|',5x,'ZETA = 0  curve',27x,'|'
     *,/,10x,'|',2x,'    3     ',2x,'|',5x,'ZETA = 0,0.5 and 1.0 curves'
     *,15x,'|'
     *,/,10x,'|',2x,'    4     ',2x,'|',5x,'ZETA = 0,0.25,0.5,0.75,1.0 c
     *urves',9x,'|')
 45   format(10x,'|',2x,'    9     ',2x,'|',5x,'EXIT to Curve Selection
     *Menu',14x,'|')
 46   format(10x,'+++++++++++++++++++++++++++++++++++++++++++++++++++++++
     *++++++++++')
 47   format(1x,///,15x,'Enter integer number for selection ===> ')
      call PSIT(19,56)
      read(*,*,err=30) ZTASEL
C
C                         *** CONSTANT ZETA CURVE SELECTION ***
      FRQCHG = 0
      if (ZTASEL .eq. 2) then
        NUMZTA = 1
        ZETA(1) = 0.
        go to 380
      else if (ZTASEL .eq. 3) then
        NUMZTA = 3
        ZETA(1) = 0.
        ZETA(2) = .5
        ZETA(3) = 1.
        go to 380
      else if (ZTASEL .eq. 4) then
        NUMZTA = 5
        ZETA(1) = 0.
        ZETA(2) = .25
        ZETA(3) = .5
        ZETA(4) = .75
        ZETA(5) = 1.
        go to 380
      else if (ZTASEL .eq. 9) then
        go to 30
      end if
C
        call CLR
        write(*,331)
```

```
 331    format(1x,////,10x,'This is the constant ZETA curve routine'
     *   ,/,10x,'----------------------------------------')
C
C     *** Enter Number of Constant ZETA Curves ***
        write(*,341)
        write(*,349)
 341    format(1x,////,10x,'How many constant ZETA curves do you wish to
     * plot? ',/)
 349    format(10x,'Enter integer value less than or equal to 10 ==> ')
        call PSIT(14,60)
        read(*,*) NUMZTA
        call CLR
        write(*,342)
 342    format(1x,///,5x,'Enter constant ZETA values in the range:  0 <=
     * ZETA <= 1.0',/,5x,'-------------------------------------------
     *--------------')
        ROW2 = 6
      do 377 I = 1,NUMZTA,1
        write(*,343) I
 343    format(1x,//,10x,'ZETA(',I2,') = ')
        ROW2 = ROW2 + 3
        call PSIT(ROW2,23)
        read(*,*) ZETA(I)
 377  continue
C
 380    call CLR
C
        FRQCHG = 0
              MAXLOG = alog10(WNMAX)
              MINLOG = alog10(WNMIN)
              INCLOG = (MAXLOG - MINLOG)/99.
              START = -99
              STOP = 0
              DO 308 I=1,NUMZTA
                START = START + 100
                STOP = STOP + 100
                if ((START.eq.1).and.((OUT.eq.1).or.(OUT.eq.5).or.
     *              (OUT.eq.6))) write (*,888)
 888    format (/////////////,25x,'* * *  Computing Data   * * *')
                if ((OUT .ge. 2) .and. (OUT .le. 4)) then
                  write(*,326) I,ZETA(I)
 326              format(1X,/,10x,'ZETA(',I2,') =',g11.4,//)
                end if
                if ((OUT .ge. 4) .and. (OUT .le. 6)) then
                  write(8,326) I,ZETA(I)
                endif
```

```
 328          R = 0.
              WNMIN2 = WNMIN
              DO 306 L = START,STOP
                B1 = 0.
                B2 = 0.
                C1 = 0.
                C2 = 0.
                D1 = 0.
                D2 = 0.
                DO 303 N=1,NUMCOF
                  K = N-1
                  IF (K) 302,301,302
 301                U = 0.0
                    U1 = -1.0
 302                U2 = 2.0*ZETA(I)*U-U1
                    D1 = (-1.0)**K*CONST(N)*WNMIN2**K*U1+D1
                    D2 = (-1.0)**K*CONST(N)*WNMIN2**K*U+D2
                    C1 = (-1.0)**K*BETA(N)*WNMIN2**K*U1+C1
                    C2 = (-1.0)**K*BETA(N)*WNMIN2**K*U+C2
                    B1 = (-1.0)**K*ALPHA(N)*WNMIN2**K*U1+B1
                    B2 = (-1.0)**K*ALPHA(N)*WNMIN2**K*U+B2
                    U1 = U
                    U = U2
 303            CONTINUE
C
                Z = 1.0E-5
                  R = R+1.
c
c . . . check for division by ~ 0
                  if (((ABS(B1*C2-B2*C1)) .le. 1.e-12) .and.
                     (L .ne. 1)) then
                    A(L) = A(L-1)
                    B(L) = B(L-1)
                    write (*,800) L
                    if ((OUT .ge. 4) .and. (OUT .le. 6)) then
                      write (8,800) L
                    end if
 800          format (5x,'Denominator approximately zero at point ',I3
     *        ,/,5x,'This point now assigned previous point value')
                    goto 802
                  else
 304                A(L) = (C1*D2-C2*D1)/(B1*C2-B2*C1)
                    B(L) = (B2*D1-B1*D2)/(B1*C2-B2*C1)
                  end if
c
c . . . check for apparent discontinuity - graphics can't handle it
```

```
                        if (L .gt. 1) then
                        if ((abs(A(L) - A(L-1)) .gt. 1.e6)
     *                     .or. (abs(B(L) - B(L-1)) .gt. 1.e6)) then
                          A(L) = A(L-1)
                          B(L) = B(L-1)
                          write (*,801) L
                          if ((OUT .ge. 4) .and. (OUT .le. 6)) then
                            write (8,801) L
                          end if
 801                      format (5x,'Apparent Discontinuity at point ',I3,
     *                  /,5x,'This point now assigned previous point value')
                        end if
                        end if
C
 802                    if (OUT .eq. 1) goto 309
                        if ((OUT .eq. 5) .or. (OUT .eq. 6)) goto 307
                        if (amod(R,10.)) 307,305,307
 305                    write(*,325) L,A(L),L,B(L)
 325                    format(11x,'A(',I3,') = ',g11.4,9x,'B(',I3,') = ',
     *                       g11.4)
 307                    if (OUT .eq. 2) goto 309
                        if ((OUT .ge. 4) .and. (OUT .le. 6)) then
                          write(8,325) L,A(L),L,B(L)
                        endif
                        if (OUT .eq. 5) goto 309
                        ALPHA1 = A(L)
                        BETA1 = B(L)
c                       call  CRVOUT (OUT,L,A(L),B(L),ALPHA1,BETA1,WN,SEL3,
c    *                           MINLOG,INCLOG,INC1,B1,B2,C1,C2,D1,D2)
                        call ROOTS (ORDER,CONST,ALPHA,BETA,NUMCOF,WNMIN,
     *                              WNMAX,ALPHA1,BETA1,FLAGRT,OUT,R)
C
 309                    continue
                        WNMIN2 = 10.**(MINLOG + R*INCLOG)
 306                continue
      if((OUT .ge. 2) .and. (OUT .le. 4)) then
        write(*,381)
 381    format(1x,///)
        PAUSE
        call CLR
      end if
C
 308  continue
C
C                            *** CONSTANT OMEGA PLOTS ***
C
```

```
      else if (SEL3 .eq. 2) then
        write(*,431)
 431    format(1x,////,10x,'This is the constant OMEGA curve routine'
     *  ,/,10x,'----------------------------------------')
C
C     *** Enter Number of Constant OMEGA Curves ***
        write(*,441)
        write(*,449)
 441    format(1x,////,10x,'How many constant OMEGA curves do you wish t
     *o plot? ',/)
 449    format(10x,'Enter integer value less than or equal to 10 ==> ')
        call PSIT(14,60)
        read(*,*) NUMWN
        call CLR
        write(*,442)
 442    format(1x,///,5x,'Enter constant OMEGA values',/,
     *                 5x,'---------------------------')
        ROW2 = 6
      do 477 I = 1,NUMWN,1
        write(*,443) I
 443    format(1x,//,10x,'OMEGA(',I2,') = ')
        ROW2 = ROW2 + 3
        call PSIT(ROW2,24)
        read(*,*) WN(I)
 477  continue
C
 480          call CLR
              INC1 = 1/99.
              STARTO = -99
              STOPO = 0
              DO 408 I=1,NUMWN
                STARTO = STARTO + 100
                STOPO = STOPO + 100
                if ((STARTO.eq.1).and.((OUT.eq.1).or.(OUT.eq.5).or.
     *              (OUT.eq.6))) write (*,888)
                if ((OUT .ge. 2) .and. (OUT .le. 4)) then
                  write(*,426) I,WN(I)
 426              format(1X,/,10x,'OMEGA(',I2,') =',g11.4,//)
                end if
                if ((OUT .ge. 4) .and. (OUT .le. 6)) then
                  write(8,426) I,WN(I)
                endif
                R = 0.
                ZETA1 = 0.
```

```
            DO 406 L = STARTO,STOPO
              B1 = 0.
              B2 = 0.
              C1 = 0.
              C2 = 0.
              D1 = 0.
              D2 = 0.
              DO 403 N=1,NUMCOF
                K = N-1
                IF (K) 402,401,402
 401              U = 0.0
                  U1 = -1.0
 402              U2 = 2.0*ZETA1*U-U1
                  D1 = (-1.0)**K*CONST(N)*WN(I)**K*U1+D1
                  D2 = (-1.0)**K*CONST(N)*WN(I)**K*U+D2
                  C1 = (-1.0)**K*BETA(N)*WN(I)**K*U1+C1
                  C2 = (-1.0)**K*BETA(N)*WN(I)**K*U+C2
                  B1 = (-1.0)**K*ALPHA(N)*WN(I)**K*U1+B1
                  B2 = (-1.0)**K*ALPHA(N)*WN(I)**K*U+B2
                  U1 = U
                  U = U2
 403          CONTINUE
C
                R = R+1.
c
c . . . check for division by ~ 0
                if ((ABS(B1*C2-B2*C1)) .le. 1.e-12) .and.
                   (L .ne. 1)) then
                  AOMEGA(L) = AOMEGA(L-1)
                  BOMEGA(L) = BOMEGA(L-1)
                  write (*,800) L
                  if ((OUT .ge. 4) .and. (OUT .le. 6)) then
                    write (8,800) L
                  end if
                  goto 803
                else
 404              AOMEGA(L) = (C1*D2-C2*D1)/(B1*C2-B2*C1)
                  BOMEGA(L) = (B2*D1-B1*D2)/(B1*C2-B2*C1)
                end if
c
c . . . check for apparent discontinuity - graphics can't handle it
                if ((abs(AOMEGA(L) - AOMEGA(L-1)) .gt. 1.e6) .or.
     *             (abs(BOMEGA(L) - BOMEGA(L-1)) .gt. 1.e6)) then
                  AOMEGA(L) = AOMEGA(L-1)
                  BOMEGA(L) = BOMEGA(L-1)
```

```
                    write (*,801) L
                    if ((OUT .ge. 4) .and. (OUT .le. 6)) then
                      write (8,801) L
                    end if
                  end if
C
 803              if (OUT .eq. 1) goto 409
                  if ((OUT .eq. 5) .or. (OUT .eq. 6)) goto 407
                  if (amod(R,10.)) 407,405,407
 405              write(*,425) L,AOMEGA(L),L,BOMEGA(L)
 425              format(11x,'AOMEGA(',I3,') = ',g11.4,9x,
     *                   'BOMEGA(',I3,') = ',g11.4)
 407              if (OUT .eq. 2) goto 409
                  if ((OUT .ge. 4) .and. (OUT .le. 6)) then
                    write(8,425) L,AOMEGA(L),L,BOMEGA(L)
                  endif
                  if (OUT .eq. 5) goto 409
                  ALPHA1 = AOMEGA(L)
                  BETA1 = BOMEGA(L)
                  call ROOTS (ORDER,CONST,ALPHA,BETA,NUMCOF,WNMIN,
     *                        WNMAX,ALPHA1,BETA1,FLAGRT,OUT,R)
C
 409              continue
                  ZETA1 = ZETA1 + INC1
 406    continue
        if((OUT .ge. 2) .and. (OUT .le. 4)) then
          write(*,381)
          PAUSE
          call CLR
        end if
C
 408    continue
C
C                          *** CONSTANT ZETA*OMEGA PLOTS ***
C
        else if (SEL3 .eq. 3) then
          write(*,531)
 531      format(1x,////,10x,'This is the constant ZETA*OMEGA curve routin
     *e'   ,/,10x,'-----------------------========================-----------')
C
C       *** Enter Number of Constant ZETA*OMEGA Curves ***
          write(*,541)
          write(*,549)
 541      format(1x,////,10x,'How many constant ZETA*OMEGA curves do you w
     *ish to plot? ',/)
 549      format(10x,'Enter integer value less than or equal to 10 ==> ')
```

```
      call PSIT(14,60)
      read(*,*) NUMZWN
      call CLR
      write(*,542)
 542  format(1x,///,5x,'Enter constant ZETA*OMEGA values',/,
    *             5x,'--------------------------------')
      ROW2 = 6
     do 577 I = 1,NUMZWN
      write(*,543) I
 543  format(1x,//,10x,'ZETA*OMEGA(',I2,') = ')
      ROW2 = ROW2 + 3
      call PSIT(ROW2,29)
      read(*,*) ZWN(I)
 577 continue
C
 580        call CLR
            MAXLOG = alog10(WNMAX)
            MINLOG = alog10(WNMIN)
            INCLOG = (MAXLOG - MINLOG)/99.
            START = -99
            STOPZO = 0
            DO 508 I=1,NUMZWN
              START = START + 100
              STOPZO = STOPZO + 100
              if ((START.eq.1).and.((OUT.eq.1).or.(OUT.eq.5).or.
    *             (OUT.eq.6))) write (*,888)
              if ((OUT .ge. 2) .and. (OUT .le. 4)) then
                write(*,526) I,ZWN(I)
 526            format(1X,/,10x,'ZETA*OMEGA(',I2,') =',g11.4,//)
              end if
              if ((OUT .ge. 4) .and. (OUT .le. 6)) then
                write(8,526) I,ZWN(I)
              endif
              R = 0.
              WNMIN2 = WNMIN
              DO 506 L = START,STOPZO
                B1 = 0.
                B2 = 0.
                C1 = 0.
                C2 = 0.
                D1 = 0.
                D2 = 0.
                DO 503 N=1,NUMCOF
                  K = N-1
                  IF (K) 502,501,502
 501                U = 0.0
```

```
                        U1 = -1.0/WNMIN2**2.
 502                    U2 = -2.0*ZWN(I)*U - WNMIN2**2*U1
                        D1 = CONST(N)*U1+D1
                        D2 = CONST(N)*U+D2
                        C1 = BETA(N)*U1+C1
                        C2 = BETA(N)*U+C2
                        B1 = ALPHA(N)*U1+B1
                        B2 = ALPHA(N)*U+B2
                        U1 = U
                        U = U2
 503                CONTINUE
                      R = R+1.
c
c . . . check for division by - 0
                      if ((ABS(B1*C2-B2*C1)) .le. 1.e-12) .and.
                         (L .ne. 1)) then
                        AZTAOM(L) = AZTAOM(L-1)
                        BZTAOM(L) = BZTAOM(L-1)
                        write (*,800) L
                        if ((OUT .ge. 4) .and. (OUT .le. 6)) then
                          write (8,800) L
                        end if
                        goto 804
                      else
 504                    AZTAOM(L) = (C1*D2-C2*D1)/(B1*C2-B2*C1)
                        BZTAOM(L) = (B2*D1-B1*D2)/(B1*C2-B2*C1)
                      end if
c
c . . . check for apparent discontinuity - graphics can't handle it
                      if ((abs(AZTAOM(L) - AZTAOM(L-1)) .gt. 1.e6) .or.
     *                   (abs(BZTAOM(L) - BZTAOM(L-1)) .gt. 1.e6)) then
                        AZTAOM(L) = AZTAOM(L-1)
                        BZTAOM(L) = BZTAOM(L-1)
                        write (*,801) L
                        if ((OUT .ge. 4) .and. (OUT .le. 6)) then
                          write (8,801) L
                        end if
                      end if
C
 804                  if (OUT .eq. 1) goto 509
                      if ((OUT .eq. 5) .or. (OUT .eq. 6)) goto 507
                      if (amod(R,10.)) 507,505,507
 505                  write(*,525) L,AZTAOM(L),L,BZTAOM(L)
 525                  format(11x,'AZTAOM(',I3,') = ',g11.4,9x,
     *                      'BZTAOM(',I3,') = 'g11.4)
 507                  if (OUT .eq. 2) goto 509
```

```
                    if ((OUT .ge. 4) .and. (OUT .le. 6)) then
                      write(8,525) L,AZTAOM(L),L,BZTAOM(L)
                    endif
                    if (OUT .eq. 5) goto 509
                    ALPHA1 = AZTAOM(L)
                    BETA1 = BZTAOM(L)
                    call ROOTS (ORDER,CONST,ALPHA,BETA,NUMCOF,WNMIN,
     *                    WNMAX,ALPHA1,BETA1,FLAGRT,OUT,R)
C
 509                continue
                    WNMIN2 = 10.**(MINLOG + R*INCLOG)
 506  continue
      if((OUT .ge. 2) .and. (OUT .le. 4)) then
        write(*,381)
        PAUSE
        call CLR
      end if
C
 508  continue
C
C           *** FREQUENCY RANGE CHANGE FOR CONSTANT ZETA PLOT ***
      else if (SEL3 .eq. 4) then
C
C     *** Enter RANGE of frequencies to examine ***
C
        write(*,296) WNMIN,WNMAX
        write(*,297)
 297    format(1x,//,5x,'Enter the RANGE of frequencies you wish to exam
     *ine',/,5x,'---------------------------------------------------',/)
        write(*,298)
 298    format(1x,/,5x,'Enter minimum frequency (WnMIN) ===> ')
        call PSIT(13,43)
        read(*,*) WNMIN
        write(*,299)
 299    format(1x,//,5x,'Enter maximum frequency (WnMAX) ===> ')
        call PSIT(16,43)
        read(*,*) WNMAX
        write(*,296) WNMIN,WNMAX
 296    format(1x,////,10x,'WNMIN =',g11.4,9x,'WNMAX =',g11.4)
        if (FILENAME .ne. '  ') then
          write(8,295) WNMIN,WNMAX
 295      format(1x,///,10x,'WNMIN =',g11.4,9x,'WNMAX ='
     *                ,g11.4,//)
        endif
        FRQCHG = 1
        SEL3 = 1
```

```
      write(*,381)
      PAUSE
C
C                          *** CALL PLOTTING ROUTINE ***
      else if (SEL3 .eq. 5) then
        SEL = 7
        FLAG = 1
      else
        continue
      end if
C
C       Recompute Alpha & Beta points in constant zeta routine when
c         frequency range is changed
      if(FRQCHG .eq. 1) go to 330
c
      if(SEL3 .eq. 5) go to 399
c
c . . . . If EXIT not selected, redisplay curve select menu
      if(SEL3 .ne. 9) go to 30
c
 399  if (FILENAME .ne. '   ') then
        close(8,STATUS='KEEP')
      endif
c
      return
      end
```

```
C
C ***********************************************************************
C                          ROOT FINDING MODULE
C ***********************************************************************
C
C
C ======================================================================
C  Subroutine  ROOTFD -- calculates the roots of the entered C. E.
C ======================================================================
C
      Subroutine ROOTS (ORDER,CONST,ALPHA,BETA,NUMCOF,WNMIN,WNMAX,
     *                 ALPHA1,BETA1,FLAGRT,OUT,R)
      real CONST(10),ALPHA(10),BETA(10),COEF(13),A,ALPHA1,BETA1,R
      real*8 B,C,P,Q,DENOM,REAL1,IMAG1,REAL2,IMAG2,IMAGSQ,
     *      REALNO,IMAGNO
      integer ORDER,I,NUMCOF,IMIN1,FLAG,FLAGRT,OUT
      character*1 CHG
      character*12 FILENAME
      common/filenam/ FILENAME
      data FLAG /0/
C
      if (FLAGRT .eq. 1) then
        call CHAREQ (ORDER,CONST,ALPHA,BETA,NUMCOF,WNMIN,WNMAX,
     *               FLAGRT,ALPHA1,BETA1)
      end if
C
        call NORMAL(ORDER,NUMCOF,CONST,ALPHA,BETA,COEF,ALPHA1,BETA1,
     *               FLAGRT)
        call ROOTS2(ORDER,COEF,FLAG,P,Q,OUT,R)
      return
      end
C
C ======================================================================
C  Subroutine  ROOTS2 -- Reduce polynomial using Bairstow's method
C ======================================================================
C
      Subroutine ROOTS2(ORDER,COEF,FLAG,P,Q,OUT,R)
      dimension F(13),G(13)
      real EPSLON,DELTAP,DELTAQ,COEF(13),R,REDUC,PREVDP,
     *     PREVDQ
      real*8 DENOM,P,Q,REAL1,IMAG1,REAL2,IMAG2,F,G
      integer J,ORDER,ORD,ORD3,N,ITERAT,MAXIT,FLAG,ORD1,OUT
      character*12 FILENAME
      logical PDIFF,QDIFF,PQCHG
      common/filenam/ FILENAME
      EPSLON = .0001
```

```
      MAXIT = 500
      ORD = ORDER
      ORD3 = ORDER + 3
      ITERAT = 1
      FLAG = 0
      P = .1
      Q = .1
      F(1) = 0.
      F(2) = 0.
      G(1) = 0.
      G(2) = 0.
      DELTAP = 0.
      DELTAQ = 0.
C
C . . . Check for roots at origin
C
      I = 0
      if (COEF(ORD3) .eq. 0.) then
 108    I = I + 1
        ORD3 = ORD3 - 1
        ORD = ORD - 1
        if ((COEF(ORD3) .eq. 0.) goto 108
        if ( ( ( OUT .eq. 3 ) .or. ( OUT .eq. 4 ) ) .and.
     *  ( amod(R,10.) .eq. 0. ) then
        write (*,107) I
        write (8,*) '  '
        write (8,107) I
 107    format(10x,I1,' root(s) at the origin')
        end if
      end if
C
      if (ORDER .eq. 1) then
        REAL1 = -COEF(4)
        IMAG1 = 0.
        if ( ( ( OUT .eq. 3 ) .or. ( OUT .eq. 4 ) ) .and.
     *  ( amod(R,10.) .eq. 0. ) then
           write (*,100) REAL1, dabs(IMAG1)
 100       format(1x,/,10x,'Only Root = ',5x,g11.4,'  +    j',g11.4)
        end if
        if ((OUT .eq. 4) .or. (OUT .eq. 6)) then
           write (8,100) REAL1, dabs(IMAG1)
        endif
        FLAG = 1
C
      else if (ORDER .eq. 2) then
        P = COEF(4)
```

```
         Q = COEF(5)
         call ROOTS1(P,Q,REAL1,IMAG1,REAL2,IMAG2)
         if (((OUT .eq. 3) .or. (OUT .eq. 4)) .and. (amod(R,10.) .eq. 0.)
     *)  then
            write (*,105) REAL1, dabs(IMAG1)
            write (*,106) REAL2, dabs(IMAG1)
            write (*,*) '  '
 105        format(10x,'First Root =  ',5x,g9.3,'  +    j',g9.3)
 106        format(10x,'Second Root = ',5x,g9.3,'  -    j',g9.3)
         end if
         if ((OUT .eq. 4) .or. (OUT .eq. 6)) then
          write (8,105) REAL1, dabs(IMAG1)
          write (8,106) REAL2, dabs(IMAG1)
          write (8,*) '  '
         endif
         FLAG = 1
C
      end if
C
C . . . If all roots computed or max iterations exceeded - finished
C
 10   if ((FLAG .eq. 1) .or. (ITERAT .gt. MAXIT)) go to 99
C
         do 110 J = 3,ORD3
           F(J) = COEF(J) - P*F(J-1) - Q*F(J-2)
           G(J) = F(J) - P*G(J-1) - Q*G(J-2)
 110     continue
         DENOM = G(ORD+1)**2 - (G(ORD+2)-F(ORD+2))*G(ORD)
C
      if (DENOM .ne. 0.) then
         PREVDP = DELTAP
         PREVDQ = DELTAQ
         DELTAP = (F(ORD+2)*G(ORD+1) - F(ORD+3)*G(ORD))*REDUC/DENOM
         DELTAQ = (G(ORD+1)*F(ORD+3) - (G(ORD+2)-F(ORD+2))*
     *            F(ORD+2))*REDUC/DENOM
         P = P + DELTAP
         Q = Q + DELTAQ
C
         if ((abs(P) .gt. EPSLON) .and. (abs(Q) .gt. EPSLON)) then
           PQCHG = (abs(DELTAP/P) + abs(DELTAQ/Q)) .lt. EPSLON
         else if  ((abs(P) .lt. EPSLON) .and. (abs(Q) .gt. EPSLON)) then
           PQCHG = (abs(DELTAP) + abs(DELTAQ/Q)) .lt. EPSLON
         else if  ((abs(P) .gt. EPSLON) .and. (abs(Q) .lt. EPSLON)) then
           PQCHG = (abs(DELTAP/P) + abs(DELTAQ)) .lt. EPSLON
         else
           PQCHG = ((abs(DELTAP) .lt. EPSLON) .and.
```

```
     *              (abs(DELTAQ) .lt. EPSLON))
        end if
        if (PQCHG) then
          call ROOTS1(P,Q,REAL1,IMAG1,REAL2,IMAG2)
            ORD1 = ORD - 1
          if (((OUT .eq. 3) .or. (OUT .eq. 4)) .and. (amod(R,10.) .eq.
     * 0.)) then
            write (*,150) ORD,REAL1,dabs(IMAG1),ORD1,REAL2,dabs(IMAG2)
 150        format(1x,'Root(',I2,') = ',g11.4,' + j',g11.4,
     *             5x,'Root(',I2,') = ',g11.4,' - j',g11.4)
          end if
          if ((OUT .eq. 4) .or. (OUT .eq. 6)) then
            write (8,150) ORD,REAL1,dabs(IMAG1),ORD1,REAL2,dabs(IMAG2)
          endif
          ORD = ORD - 2
C
C . . . Check order of reduced polynomial
          if (ORD .eq. 0) then
            FLAG = 1
          else if (ORD .eq. 1) then
            REAL1 = -F(ORD+3)/F(ORD+2)
            IMAG1 = 0.
            if (((OUT .eq. 3) .or. (OUT .eq. 4)) .and. (amod(R,10.) .eq.
     *        0.)) then
              write (*,160) ORD,REAL1,IMAG1
 160          format(1x,'Root(',I2,') = ',g11.4,' + j',g11.4,//)
            end if
            if ((OUT .eq. 4) .or. (OUT .eq. 6)) then
              write (8,160) ORD,REAL1,IMAG1
            endif
            FLAG = 1
          else
            ORD3 = ORD + 3
            do 180 J = 3,ORD3
              COEF(J) = F(J)
              ITERAT = 1
 180        continue
          end if
C
        else
          ITERAT = ITERAT + 1
        end if
C
      else
        P = P + 1
        Q = Q + 1
```

```
          ITERAT = 1
        end if
C
        PDIFF = (abs(PREVDP) + abs(DELTAP)) .ne. abs(PREVDP + DELTAP)
        QDIFF = (abs(PREVDQ) + abs(DELTAQ)) .ne. abs(PREVDQ + DELTAQ)
        if ((PDIFF .and. (Q .lt. .01)) .or. (QDIFF .and. (P .lt. .01))
     *    .or. (PDIFF .and. QDIFF)) then
          REDUC = REDUC/2.
        end if
        if (FLAG .ne. 1) go to 10
 99     return
        end
C
C ======================================================================
C  Subroutine  ROOTS1 -- Finds roots of second order polynomial
C ======================================================================
C
        Subroutine ROOTS1(B,C,REAL1,IMAG1,REAL2,IMAG2)
        real*8 B,C,REAL1,IMAG1,REAL2,IMAG2,REALNO,IMAGNO,IMAGSQ
C
        REALNO = -B/2.
        IMAGNO = B**2 - 4*C
        if (IMAGNO .ge. 0.) then
          IMAGSQ = dsqrt(IMAGNO)/2.
          REAL1 = REALNO + IMAGSQ
          IMAG1 = 0.
          REAL2 = REALNO - IMAGSQ
          IMAG2 = 0.
        else
          IMAGSQ = dsqrt(-IMAGNO)/2.
          REAL1 = REALNO
          IMAG1 = IMAGSQ
          REAL2 = REALNO
          IMAG2 = -IMAGSQ
        end if
C
        return
        end
C
C ======================================================================
C  Subroutine  NORMAL -- Adds CONST, ALPHA & BETA terms and normalizes
C ======================================================================
C
        Subroutine NORMAL(ORDER,NUMCOF,CONST,ALPHA,BETA,COEF,ALPHA1,BETA1,
     *                   FLAGRT)
        real CONST(10),ALPHA(10),BETA(10),COEF(10),ACOEF,BCOEF,CCOEF,
```

```
     *     ALPHA1,BETA1
      integer J,ORDER,NUMCOF,N,M,FLAGRT
C
      N = 3
      M = ORDER + 3
      do 10 J = NUMCOF,1,-1
        COEF(N) = CONST(J) + ALPHA(J)*ALPHA1 + BETA(J)*BETA1
        N = N + 1
 10   continue
      if (COEF(NUMCOF) .ne. 1.) then
        do 20 J = 4,M
          COEF(J) = COEF(J)/COEF(3)
 20     continue
      COEF(3) = 1.
      end if
      if (FLAGRT .eq. 1) then
      call CLR
      I = NUMCOF + 1
      do 25 J = 3,M
      I = I - 1
      write (*,30) I,COEF(J)
 30   format(15x,'COEF(',I2,') = ',g9.3,/)
 25   continue
      end if
      return
      end
```

```
C
C ************************************************************************
C                          PLOTTING MODULE
C ************************************************************************
C
C
C ========================================================================
C  Subroutine  RUNPRO -- executes the parameter plane program
C ========================================================================
C
      Subroutine RUNPRO (NUMZTA,STOP,ZETA,CURV,NUMWN,WN,STOPO,
     *                   ZWN,NUMZWN,STOPZO)
      real ATEMP(102),BTEMP(102),ZETA(10),A,B,AOMEGA,BOMEGA,
     *     XMIN,XMAX,YMIN,YMAX,FACT,XCEN,YCEN,WN(10),ZWN(10),
     *     EXPAMN,EXPAMX,EXPBMN,EXPBMX,AZTAOM,BZTAOM,
     *     EXPALP(1002),EXPBET(1002),STITLE,CRVTYP,
     *     AZFST,AZDLT,BZFST,BZDLT,AOFST,AODLT,BOFST,BODLT,
     *     AZOFST,AZODLT,BZOFST,BZODLT
      integer I,J,IOPORT,MODEL,NUMZTA,STOP,FRSTZ,DELTZ,INTRVL,CURV,
     *        DELT1,DELT2,NUMWN,L,K,STOPO,FRSTW,DELTW,EXPAND,INTSRT,
     *        INTSTP,NC,NUMZWN,STOPZO,SEL,SYMBL
      dimension A(1002),B(1002),AOMEGA(1002),BOMEGA(1002),AZTAOM(1002),
     *          BZTAOM(1002)
      character*1 CHG,NMCHAR(30)
      character*30 TITLE
      common A,B,AOMEGA,BOMEGA,AZTAOM,BZTAOM
      common/plottr/ IOPORT,MODEL
      common/box/ EXPAND,SYMBL
      common/factr/ FACT
c     common/symb/ SYMBL
      equivalence (NMCHAR(1),TITLE)
C
      FACT = 0.9
      EXPAND = 9
      SYMBL = 14
C
 78   call CLR
      write(*,80)
      write(*,81)
      write(*,82)
      write(*,81)
      write(*,83)
      write(*,81)
      write(*,93)
 80   format(1x,///////,10x,'-----------------------------------------
     *---------------------',/,10x,'|',24x,'PLOTTING MENU',25x,'|')
```

```
 81    format(10x,'--------------------------------------------------------
      *-----------')
 82    format(10x,'|',2x,'OPTION NO.',2x,'|',20x,'OPTION',21x,'|')
 83    format(10x,'|',2x,'     1      ',2x,'|',5x,
      *'TITLE output graph and PLOT data',10x,'|'
      *,/,10x,'|',2x,'     2      ',2x,'|',5x,
      *'PLOT data (no title)',22x,'|'
      *,/,10x,'|',2x,'     3      ',2x,'|',5x,
      *'SIZE output graph',25x,'|'
      *,/,10x,'|',2x,'     4      ',2x,'|',5x,
      *'SYMBOL to be plotted at each data point',3x,'|'
      *,/,10x,'|',2x,'     9      ',2x,'|',5x,'EXIT to Main Menu',25x,'|')
 93    format(1x,///,15x,'Enter integer number for selection ===> ')
       call PSIT(23,56)
       read(*,*,err=78) SEL
       if (SEL .eq. 1) then
           goto 4
         else if (SEL .eq. 2) then
           goto 9
         else if (SEL .eq. 3) then
           goto 7
         else if (SEL .eq. 4) then
           goto 5
         else if (SEL .eq. 9) then
           goto 399
         else
           goto 78
       end if
C
C. . .  Adjust size of output plot
 7     call CLR
       write(*,15)
 15    format(/////,10x,'Output plots are currently sized to fill the con
      *sole screen.',//,
      *       10x,'Would you like to adjust the plot size?',/,10x,
      *       'Enter "y" or "n" ==> ')
       call PSIT(10,31)
       read(*,'(A)',err=4) CHG
       if ((CHG .eq. 'n') .or. (CHG .eq. 'N')) then
         go to 78
       end if
       write(*,16)
 16    format(///,10x,'A default plot factor size of 0.9 is used to fill
      *the console screen.',/,10x,
      *'This plot size will be halved by entering a value of 0.45.',/,
      *10x,'It will be doubled with a factor entry of 1.8.',//,10x,
```

```
     *'Enter factor size ==>  ')
      call PSIT(18,32)
      read(*,*) FACT
      goto 78
C
C. . .  Select symbol to be plotted at each data point
  5   call CLR
      write(*,17)
 17   format(/////,10x,'Type in an INTEGER number from 0 through 13 to p
     *lace a',/,
     * 10x,'symbol at each calculated data point. There are 100 data poi
     *nts',/,
     * 10x,'for each curve. Examples of symbols with associated numbers:
     *',//,
     * 15x,'2   Triangle',/,
     * 15x,'3   +',/,
     * 15x,'4   X',/,
     * 15x,'8   Z',/,
     * 15x,'9   Y',/,
     * 15x,'11  *',/,
     * 15x,'13  Vertical Line',///,
     * 10x,'Enter INTEGER number (0 - 13) ==> ')
      call PSIT(20,45)
      read(*,*,err=78) SYMBL
        go to 78
C
C . . . Enter TITLE of graph
 4    call CLR
      write(*,6)
 6    format(/////,10x,'Enter plot title ==>  ',/,
     *10x,'(30 characters max)')
      call PSIT(7,31)
      read(*,8) TITLE
 8    format(A30)
      do 11 I = 30,1,-1
        if (NMCHAR(I) .ne. ' ') go to 13
 11   continue
 13   NC = I
      STITLE = 4.3 - .12*NC
C
 9    call CLR
      INTRVL = 0
      SEL = 3
      CURV = 0
      call PLOTS (0,IOPORT,MODEL)
      call NEWPEN (1)
```

```
C
C ..... Calculate Constant ZETA Plot
C
      if (NUMZTA .ge. 1) then
        CRVTYP = .1
        write (*,119)
 119    format(///,25x,'CONSTANT ZETA CURVES')
        call PLTCRV (STOP,A,B,NUMZTA,CRVTYP,ZETA,
     *               TITLE,STITLE,NC,AZFST,AZDLT,BZFST,BZDLT)
        call PLOT (0.0,0.0,-999)
      end if
C
C ..... Calculate Constant OMEGA Plot
C
      if (NUMWN .ge. 1) then
        call CLR
        CRVTYP = -.1
        write (*,219)
 219    format(///,25x,'CONSTANT OMEGA CURVES')
        call PLTCRV (STOPO,AOMEGA,BOMEGA,NUMWN,CRVTYP,WN,
     *               TITLE,STITLE,NC,AOFST,AODLT,BOFST,BODLT)
        call PLOT (0.0,0.0,-999)
      end if
C
C ..... Calculate Constant ZETA*OMEGA Plot
C
      if (NUMZWN .ge. 1) then
        call CLR
        CRVTYP = -.2
        write (*,319)
 319    format(///,25x,'CONSTANT ZETA*OMEGA CURVES')
        call PLTCRV (STOPZO,AZTAOM,BZTAOM,NUMZWN,CRVTYP,ZWN,
     *               TITLE,STITLE,NC,AZOFST,AZODLT,BZOFST,BZODLT)
        call PLOT (0.0,0.0,-999)
      end if
C
C ... Calculate Zeta & Omega Plot
C
      if ((NUMZTA .ge. 1) .and. (NUMWN .ge. 1)) then
      call CLR
      ATEMP(101) = AZFST
      ATEMP(102) = AZDLT
      BTEMP(101) = BZFST
      BTEMP(102) = BZDLT
      write (*,492)
 492  format(///,25x,'CONSTANT COMBINED CURVES')
```

```
      write (*,493) ATEMP(101),ATEMP(102)
 493  format (///,5x,'X Axis Min Value = ',g11.4,5x,'X Axis DELTA = ',
     *        g11.4,//)
      write (*,494) BTEMP(101),BTEMP(102)
 494  format (5x,'Y Axis Min Value = ',g11.3,5x,'Y Axis DELTA = ',g11.4,
     *        /////)
      write (*,495)
 495  format (25x,'*** Calculating Data ***',////)
c
      INTRVL = 0
      call FACTOR (FACT)
      call ASPECT (1.2)
      call STAXIS (.13,.20,.15,0.1,2)
      call SYMBOL (STITLE,6.0,.25,TITLE,0.,NC)
      call AXIS (.8,.8,'Alpha',-5,-7.0,0.,AZFST,AZDLT)
      call AXIS (.8,.8,'Beta',4,-5.0,90.,BZFST,BZDLT)
C
C . . . Draw dashed lines at axis increments and surrounding box
      call GRID
C
      do 470 I = 1,NUMWN
        do 480 J = 1,100
          ATEMP(J) = AOMEGA(J + INTRVL)
          BTEMP(J) = BOMEGA(J + INTRVL)
 480    continue
          call CURVE (ATEMP,BTEMP,100,-.1)
          call WHERE (X,Y,FACT)
          call NUMBER (X,Y,0.15,WN(I),0.,2)
          INTRVL = INTRVL + 100
 470  continue
c
      INTRVL = 0
      do 440 I = 1,NUMZTA
        do 450 J = 1,100
          ATEMP(J) = A(J + INTRVL)
          BTEMP(J) = B(J + INTRVL)
 450    continue
          call CURVE (ATEMP,BTEMP,100,.1)
          call WHERE (X,Y,FACT)
          call NUMBER (X,Y,0.15,ZETA(I),0.,2)
          INTRVL = INTRVL + 100
 440  continue
c
      if (NUMZWN .ge. 1) then
      INTRVL = 0
      do 540 I = 1,NUMZWN
```

```
        do 550 J = 1,100
          ATEMP(J) = AZTAOM(J + INTRVL)
          BTEMP(J) = BZTAOM(J + INTRVL)
 550    continue
          call CURVE (ATEMP,BTEMP,100,-.2)
          call WHERE (X,Y,FACT)
          call NUMBER (X,Y,0.15,ZWN(I),0.,2)
          INTRVL = INTRVL + 100
 540  continue
      end if
      end if
      call PLOT (0.0,0.0,999)
C
C . . . Expand plot about a selected area or point
C
 30   call CLR
      write(*,31)
      write(*,32)
      write(*,33)
      write(*,32)
      write(*,34)
      write(*,36)
      write(*,35)
      write(*,32)
      write(*,37)
 31   format(1x,//,10x,'---------------------------------------------------
     *----------------',/,10x,'|',19x,'EXPAND PLOT MENU',27x,'|')
 32   format(10x,'|---------------------------------------------------
     *---------|')
 33   format(10x,'|',2x,'OPTION NO.',2x,'|',14x,'EXPANSION SELECTION'
     *,14x,'|')
 34   format(10x,'|',2x,'    1     ',2x,'|',5x,'Expand area defined by a
     *xes values',8x,'|'
     *,/,10x,'|',2x,'    2     ',2x,'|',5x,'Expand around a selected poi
     *nt',12x,'|')
 35   format(10x,'|',2x,'    9     ',2x,'|',5x,'EXIT plotting routine'
     *,21x,'|')
 36   format(10x,'|+++++++++++++++++++++++++++++++++++++++++++++++++++
     *+++++++++|')
 37   format(1x,///,15x,'Enter integer number for selection ===> ')
      call PSIT(17,56)
      read(*,*,err=30) EXPAND
C
 294  if (EXPAND .eq. 2) then
C
```

```
C . . . Expand plot around selected point
C
      call CLR
      write(*,295)
 295  format(1x,//,5x,'Enter the POINT about which you wish to expand',
     */,5x,'--------------------------------------------',/,5x,'(This
     *point must be entered in # of divisions on alpha & beta',/,5x,'axe
     *s. An expansion about the center of the diagram would ',/,5x,'requ
     *ire an entry of 3.5 for the alpha axis and 2.5 for the ',/,5x,'bet
     *a axis.)',//)
      write(*,297)
 297  format(1x,//,5x,'Enter CENTER value for ALPHA axis (range is 0 to
     *7) ===> ')
      call PSIT(14,63)
      read(*,*) XCEN
      write(*,298)
 298  format(1x,//,5x,'Enter CENTER value for BETA axis (range is 0 to 5
     *) ===> ')
      call PSIT(17,62)
      read(*,*) YCEN
      write(*,300)
 300  format(1x,//,5x,'Enter value for FACTOR (enlargement) ===> ')
      call PSIT(20,50)
      read(*,*) FACT
c
c     Compute axis scaling values
c
      XCEN = XCEN * FACT
      YCEN = YCEN * FACT
c
      XMIN = XCEN - 3.5
      XMAX = XCEN + 3.5
      YMIN = YCEN - 2.5
      YMAX = YCEN + 2.5
c
      if (NUMZTA .eq. 0) then
          ATEMP(101) = AOFST
          ATEMP(102) = AODLT
          BTEMP(101) = BOFST
          BTEMP(102) = BODLT
      else
          ATEMP(101) = AZFST
          ATEMP(102) = AZDLT
          BTEMP(101) = BZFST
          BTEMP(102) = BZDLT
      end if
```

```
      call CLR
      write (*,392) ATEMP(101),ATEMP(102)
 392  format (/////,5x,'X Axis Min Value = ',g11.4,5x,'X Axis DELTA = ',
     *        g11.4,//)
      write (*,393) BTEMP(101),BTEMP(102)
 393  format (5x,'Y Axis Min Value = ',g11.3,5x,'Y Axis DELTA = ',g11.4,
     *        /////)
c
      call  PLOTS (0,IOPORT,MODEL)
      call WINDOW (XMIN,YMIN,XMAX,YMAX)
      call FACTOR (FACT)
      call ASPECT (1.2)
      call STAXIS (.13,.20..15,0.1,2)
      call SYMBOL (STITLE,6.0,.25,TITLE,0.,NC)
      call AXIS (.8,.8,'Alpha',-5,-7.0,0.,ATEMP(101),ATEMP(102))
      call AXIS (.8,.8,'Beta',4,-5.0,90.,BTEMP(101),BTEMP(102))
      INTRVL = 0
      write (*,331)
 331  format (///,25x,'*** Calculating Data ***',//)
C
C . . . Draw dashed lines at axis increments and surrounding box
      call GRID
C
      do 340 I = 1,NUMZTA
        do 350 J = 1,100
          ATEMP(J) = A(J + INTRVL)
          BTEMP(J) = B(J + INTRVL)
 350    continue
          call CURVE (ATEMP,BTEMP,100,.1)
          call WHERE (X,Y,FACT)
          call NUMBER (X,Y,0.15,ZETA(I),0.,2)
         if ((SYMBL .le. 13) .and. (SYMBL .ge. 0)) then
            call LINE (ATEMP,BTEMP,100,1,-1,SYMBL)
         end if
          INTRVL = INTRVL + 100
 340      continue
C
      INTRVL = 0
      do 370 I = 1,NUMWN
        do 380 J = 1,100
          ATEMP(J) = AOMEGA(J + INTRVL)
          BTEMP(J) = BOMEGA(J + INTRVL)
 380    continue
          call CURVE (ATEMP,BTEMP,100,-.1)
          call WHERE (X,Y,FACT)
          call NUMBER (X,Y,0.15,WN(I),0.,2)
```

```
          if ((SYMBL .le. 13) .and. (SYMBL .ge. 0)) then
             call LINE (ATEMP,BTEMP,100,1,-1,SYMBL)
          end if
 381       continue
           INTRVL = INTRVL + 100
 370  continue
      call PLOT (0.0,0.0,999)
      FACT = 0.9
C
C
C . . . Enter RANGE of ALPHA & BETA values to examine
C
      else if (EXPAND .eq. 1) then
        call CLR
 700    write(*,701)
 701    format(////,10x,'Enter Minimum ALPHA axis value  ==>  ')
        call PSIT(6,48)
        read(*,*,err=700) EXPAMN
 702    write(*,703)
 703    format(/,10x,'Enter Maximum ALPHA axis value  ==>  ')
        call PSIT(8,48)
        read(*,*,err=702) EXPAMX
 704    write(*,705)
 705    format(/,10x,'Enter Minimum BETA axis value  ==>  ')
        call PSIT(10,47)
        read(*,*,err=700) EXPBMN
 706    write(*,707)
 707    format(/,10x,'Enter Maximum BETA axis value  ==>  ')
        call PSIT(12,47)
        read(*,*,err=702) EXPBMX
C
C ..... Calculate Constant ZETA Plot
        if (NUMZTA .ge. 1) then
          J = 0
          do 710 I = 1,STOP
            if((A(I) .ge. EXPAMN) .and. (A(I) .le. EXPAMX)) then
              if((B(I) .ge. EXPBMN) .and. (B(I) .le. EXPBMX)) then
                J = J + 1
                EXPALP(J) = A(I)
                EXPBET(J) = B(I)
              end if
            end if
 710      continue
          FRSTZ = J + 1
          DELTZ = J + 2
C
```

```
          call SCALE (EXPALP,7.0,J,1)
          call SCALE (EXPBET,5.0,J,1)
C
          write (*,711)
 711      format(///////,30x,'WINDOW EXPANSION')
        write (*,712) EXPALP(FRSTZ),EXPALP(DELTZ)
 712    format (/////,5x,'X Axis Min Value = ',g11.4,5x,'X Axis DELTA = ',
       *        g11.4,//)
        write (*,714) EXPBET(FRSTZ),EXPBET(DELTZ)
 714    format (5x,'Y Axis Min Value = ',g11.3,5x,'Y Axis DELTA = ',g11.4,
       *        /////)
        write (*,716)
 716    format (///,25x,'*** Calculating Data ***',//)
C
          call PLOTS (0,IOPORT,MODEL)
          call FACTOR (FACT)
          call NEWPEN (1)
          call ASPECT (1.2)
          call STAXIS (.13,.20,.15,0.1,2)
          call SYMBOL (STITLE,6.0,.25,TITLE,0.,NC)
          call AXIS (.8,.8,'Alpha',-5,-7.0,0.,EXPALP(FRSTZ),EXPALP(DELTZ))
          call AXIS (.8,.8,'Beta',4,-5.0,90.,EXPBET(FRSTZ),EXPBET(DELTZ))
C
C . . . Draw dashed lines at axis increments and surrounding box
          call GRID
C
          INTSRT = 1
          INTSTP = 100
          do 720 K = 1,NUMZTA
          J = 0
          J1 = 0
            do 730 I = INTSRT,INTSTP
              if((A(I) .ge. EXPAMN) .and. (A(I) .le. EXPAMX)) then
                if((B(I) .ge. EXPBMN) .and. (B(I) .le. EXPBMX)) then
                  J = J + 1
                  ATEMP(J) = A(I)
                  BTEMP(J) = B(I)
                end if
              end if
 730        continue
            ATEMP(J+1) = EXPALP(FRSTZ)
            ATEMP(J+2) = EXPALP(DELTZ)
            BTEMP(J+1) = EXPBET(FRSTZ)
            BTEMP(J+2) = EXPBET(DELTZ)
            call CURVE (ATEMP,BTEMP,J,.1)
           if ((SYMBL .le. 13) .and. (SYMBL .ge. 0)) then
```

```
          call LINE (ATEMP,BTEMP,J,1,-1,SYMBL)
        end if
         J1 = J-1
         call WHERE(X,Y,FACT)
         call NUMBER (X,Y,0.15,ZETA(K),0.,2)
         INTSRT = INTSRT + 100
         INTSTP = INTSTP + 100
 720     continue
       end if
C
C . . . Constant Wn curves
C
       if (NUMWN .ge. 1) then
       if (NUMZTA .ge. 1) go to 819
         J = 0
         do 810 I = 1,STOPO
           if((AOMEGA(I) .ge. EXPAMN) .and. (AOMEGA(I) .le. EXPAMX))
     *     then
             if((BOMEGA(I) .ge. EXPBMN) .and. (BOMEGA(I) .le. EXPBMX))
     *       then
               J = J + 1
               EXPALP(J) = AOMEGA(I)
               EXPBET(J) = BOMEGA(I)
             end if
           end if
 810     continue
         FRSTZ = J + 1
         DELTZ = J + 2
C
       call SCALE (EXPALP,7.0,J,1)
       call SCALE (EXPBET,5.0,J,1)
C
       write (*,811)
 811   format(///////,30x,'WINDOW EXPANSION')
      write (*,812) EXPALP(FRSTZ),EXPALP(DELTZ)
 812  format (/////,5x,'X Axis Min Value = ',g11.4,5x,'X Axis DELTA = ',
     *        g11.4,//)
      write (*,814) EXPBET(FRSTZ),EXPBET(DELTZ)
 814  format (5x,'Y Axis Min Value = ',g11.3,5x,'Y Axis DELTA = ',g11.4,
     *        /////)
      write (*,816)
 816  format (///,25x,'*** Calculating Data ***',//)
C
       call PLOTS (0,IOPORT,MODEL)
       call FACTOR (0.9)
       call NEWPEN (1)
```

```
        call ASPECT (1.2)
        call STAXIS (.13,.20,.15,0.1,2)
        call SYMBOL (STITLE,6.0,.25,TITLE,0.,NC)
        call AXIS (.8,.8,'Alpha',-5,-7.0,0.,EXPALP(FRSTZ),EXPALP(DELTZ))
        call AXIS (.8,.8,'Beta',4,-5.0,90.,EXPBET(FRSTZ),EXPBET(DELTZ))
C
C . . . Draw dashed lines at axis increments and surrounding box
        call GRID
C
 819    INTSRT = 1
        INTSTP = 100
        do 820 K = 1,NUMWN
        J = 0
        J1 = 0
          do 830 I = INTSRT,INTSTP
            if((AOMEGA(I) .ge. EXPAMN) .and. (AOMEGA(I) .le. EXPAMX))
     *      then
              if((BOMEGA(I) .ge. EXPBMN) .and. (BOMEGA(I) .le. EXPBMX))
     *        then
                J = J + 1
                ATEMP(J) = AOMEGA(I)
                BTEMP(J) = BOMEGA(I)
              end if
            end if
 830      continue
          ATEMP(J+1) = EXPALP(FRSTZ)
          ATEMP(J+2) = EXPALP(DELTZ)
          BTEMP(J+1) = EXPBET(FRSTZ)
          BTEMP(J+2) = EXPBET(DELTZ)
          call CURVE (ATEMP,BTEMP,J,-.1)
         if ((SYMBL .le. 13) .and. (SYMBL .ge. 0)) then
            call LINE (ATEMP,BTEMP,J,1,-1,SYMBL)
         end if
        call WHERE(X,Y,FACT)
          call NUMBER (X,Y,0.15,WN(K),0.,2)
          J1 = J-1
          INTSRT = INTSRT + 100
          INTSTP = INTSTP + 100
 820    continue
        end if
        call PLOT (0.0,0.0,999)
      else
        go to 399
      end if
C
C
```

```
      call CLR
      write(*,239)
 239  format(1x,///////,9x,'Do you want to make another WINDOW selection
     *?',//,9x,'Enter "y" or "n" ==> ')
      call PSIT(11,31)
      read(*,'(A)',err=399) CHG
      if ((CHG .eq. 'y') .or. (CHG .eq. 'Y')) then
        go to 30
      end if
 399  call CLR
      write(*,238)

 238  format(1x,///////,9x,'Do you want to SAVE your constant CURVE sele
     *ctions?',//,9x,'Enter "y" or "n" ==> ')
      call PSIT(11,31)
      read(*,'(A)',err=399) CHG
      if ((CHG .eq. 'n') .or. (CHG .eq. 'N')) then
        NUMZTA = 0
        NUMWN = 0
        NUMZWN = 0
      end if
      return
      end
C
C ======================================================================
C  Subroutine  GRID -- draws dashed lines at axis increments
C ======================================================================
C
      Subroutine GRID
      integer EXPAND,SYMBL
      common/box/ EXPAND,SYMBL
C
        if (EXPAND .eq. 2) then
          call PLOT (0.8,0.8,-3)
        else
          call PLOT (0.8,0.8,-3)
        endif
        call PLOT (0.0,5.0,3)
        call PLOT (7.0,5.0,2)
        call PLOT (7.0,0.0,2)
        call STDASH (.01,.10)
        call PLOTD (1.0,0.0,3)
        call PLOTD (1.0,5.0,2)
        call PLOTD (2.0,5.0,3)
        call PLOTD (2.0,0.0,2)
        call PLOTD (3.0,0.0,3)
```

```
        call PLOTD (3.0,5.0,2)
        call PLOTD (4.0,5.0,3)
        call PLOTD (4.0,0.0,2)
        call PLOTD (5.0,0.0,3)
        call PLOTD (5.0,5.0,2)
        call PLOTD (6.0,5.0,3)
        call PLOTD (6.0,0.0,2)
        call PLOTD (7.0,1.0,3)
        call PLOTD (0.0,1.0,2)
        call PLOTD (0.0,2.0,3)
        call PLOTD (7.0,2.0,2)
        call PLOTD (7.0,3.0,3)
        call PLOTD (0.0,3.0,2)
        call PLOTD (0.0,4.0,3)
        call PLOTD (7.0,4.0,2)
C
      return
      end
C
C ======================================================================
C  Subroutine  PLTCRV -- calculates curves based on input arrays
C ======================================================================
C
      Subroutine PLTCRV (STP,APTS,BPTS,NUMPTS,CRVTYP,CRVNUM,
     *                 TITLE,STITLE,NC,AFIRST,ADELTA,BFIRST,BDELTA)
      real ATEMP(102),BTEMP(102),APTS(1002),BPTS(1002),CRVNUM(10),
     *     STITLE,CRVTYP,AFIRST,ADELTA,BFIRST,BDELTA
      integer STP,FRST,DELT,INTRVL,NC,NUMPTS,EXPAND,SYMBL
      character*30 TITLE
      common/box/ EXPAND,SYMBL
      common/factr/ FACT
c     common/symb/ SYMBL
C
      call FACTOR (FACT)
      call ASPECT (1.2)
      call STAXIS (.13,.20,.15,0.1,2)
        INTRVL = 0
        FRST = STP + 1
        DELT = STP + 2
        call SCALE (APTS,7.0,STP,1)
        ATEMP(101) = APTS(FRST)
        ATEMP(102) = APTS(DELT)
        AFIRST = APTS(FRST)
        ADELTA = APTS(DELT)
        write (*,120) ATEMP(101),ATEMP(102)
 120    format (///,5x,'X Axis Min Value = ',g11.4,5x,'X Axis DELTA = ',
```

```
     *        g11.4,//)
          call SCALE (BPTS,5.0,STP,1)
          BTEMP(101) = BPTS(FRST)
          BTEMP(102) = BPTS(DELT)
          BFIRST = BPTS(FRST)
          BDELTA = BPTS(DELT)
          write (*,130) BTEMP(101),BTEMP(102)
 130      format (5x,'Y Axis Min Value = ',g11.3,5x,'Y Axis DELTA = ',
     *         g11.4,/////)
          write (*,131)
 131      format (25x,'*** Calculating Data ***',////)
          call AXIS (.8,.8,'Alpha',-5,-7.0,0.,APTS(FRST),APTS(DELT))
          call AXIS (.8,.8,'Beta',4,-5.0,90.,BPTS(FRST),BPTS(DELT))
          call SYMBOL (STITLE,6.0,.25,TITLE,0.,NC)
C
C . . . Draw dashed lines at axis increments and surrounding box
          call GRID
C
          do 140 I = 1,NUMPTS
            do 150 J = 1,100
              ATEMP(J) = APTS(J + INTRVL)
              BTEMP(J) = BPTS(J + INTRVL)
 150        continue
            call CURVE (ATEMP,BTEMP,100,CRVTYP)
            call WHERE (X,Y,FACT)
              call NUMBER (X,Y,0.18,CRVNUM(I),0.,2)
            if ((SYMBL .le. 13) .and. (SYMBL .ge. 0)) then
              call LINE (ATEMP,BTEMP,100,1,-1,SYMBL)
            end if
c           do 161 K = 1,100,99
c             ATEMP(K) = ATEMP(K) / APTS(DELT)
c             BTEMP(K) = BTEMP(K) / BPTS(DELT)
c             call NUMBER (ATEMP(K),BTEMP(K),0.18,CRVNUM(I),0.,2)
c161        continue
            INTRVL = INTRVL + 100
 140      continue
C
        return
        end
```

```
C
C ********************************************************************
C                               ASCII MODULE
C ********************************************************************
C
C ====================================================================
C  Subroutine CLR -- clears the screen
C ====================================================================
C
      Subroutine CLR
      character*1 C1, C2, C3, C4
      integer IC(4)
      equivalence (C1,IC(1)),(C2,IC(2)),(C3,IC(3)),(C4,IC(4))
      data IC/16#1B,16#5B,16#32,16#4A/
      write(*,1) C1,C2,C3,C4
1     format(1X,4A1)
      return
      end
C
C ====================================================================
C  Subroutine PSIT -- positions cursor by row and column
C ====================================================================
C
      Subroutine PSIT(ROW,COLUMN)
      integer IC(4),ROW,COLUMN,L
      character*1 C1,C2,C5,C8,LC(5)
      character*5 CBUFF
      equivalence (C1,IC(1)),(C2,IC(2)),(C5,IC(3)),(C8,IC(4)),
     *            (CBUFF,LC(1))
      data IC/16#1B,16#5B,16#3B,16#66/
C
      L=10000+100*ROW+COLUMN
C
C +++ Write Escape Codes to a Character Buffer +++
      write(CBUFF,2) L
    2 format(I5)
C
C +++ Write Escape Codes to Display +++
      write(*,3) C1,C2,LC(2),LC(3),C5,LC(4),LC(5),C8
    3 format(1X,8A1,\)
      return
      end
```

LIST OF REFERENCES

1. Mitrovic, D., "Graphical Analysis and Synthesis of Feedback Control Systems", Part I, Theory and Analysis, Part II, Synthesis, AIEE Transactions, Part II, January 1959, pp. 476-496.

2. Siljak, D.D., "Analysis and Synthesis of Feedback Control Systems in the Parameter Plane", Part I, Linear Continuous Systems, IEEE Transactions, 4 November 1964, pp. 449-458.

3. Thaler, G.J. and Towill, D.R. "Parameter Space Methods for Dynamic Systems, "Unpublished notes, EC 4370.

4. Thaler, G.J. and Ohta, T., Mitrovic's Method-Some Fundamental Techniques, Research Paper No. 42, Naval Postgraduate School, Monterey, CA, January 1964.

5. Nutting, R.M., Parametaer Plane Techniques for Feedback Control Systems by Applying Mitrovic's Method, Master's Thesis, Naval Postgraduate School, Monterey, CA., 1965.

6. Potter, D. M., Feedback Control Analysis Using Parameter Plane Techniques, Master's Thesis, Naval Postgraduate School, Monterey, CA., June 1986.

7. Choe, H.H., Some Extensions of Mitrovic's Method in Analysis and Design of Feedback Control Systems, Master's Thesis, Naval Postgraduate School, Monterey, CA., 1964.

8. Hyon, C.H., A Direct Method of Designing Linear Feedback Control Systems by Applying Mitrovic's Method, Master's Thesis, Naval Postgraduate School, Monterey, CA., 1964.

9. Wood, R.L. Jr., Microcomputer Based Linear System Design Tool, Master's Thesis, Naval Postgraduate School, Monterey, CA., September 1986.

10. Young, T.L. and Van Woert, M.L., PLOT88 Software Library Reference Manual, Plotworks Inc., January 1984.

INITIAL DISTRIBUTION LIST

	No. Copies
1. Defense Technical Information Center Cameron Station Alexandria, VA 22304-6145	2
2. Library, Code 0142 Naval Postgraduate School Monterey, CA 93943-5002	2
3. Chairman, Code 62 Department of Electrical and Computer Engineering Naval Postgraduate School Monterey, CA 93943-5000	1
4. Professor G.J. Thaler, Code 62Tr Department of Electrical and Computer Engineering Naval Postgraduate School Monterey, Ca 93943-5000	5
5. Professor H. Titus, Code 62Ts Department of Electrical and Computer Engineering Naval Postgraduate School Monterey, CA 93943-5000	1
6. LCDR S.S. Lee Department of Electrical Engineering Naval Academy, Jinhae City Gyungnam 602-00 Republic of Korea	1
7. LCDR R.J. Kranz 1740 Dietz Place NW Albuquerque, NM 87107	2
8. Captain Roger Nutting, Director Ship Design Group, Code SEA-50 Naval Sea Systems Command Washington, D.C. 20362-5101	1
9. Mr. Dean Carico, Code RW-042 RWATD NATC Putuxent River, MD. 20670	1

Thesis
K8515 Kranz
c.1 Parameter plane analysis of automatic control systems using an IBM compatible microcomputer.